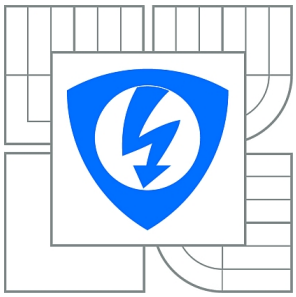
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF MICROELECTRONICS

GRAFICKÝ KONTROLÉR PRO OBVODY FPGA

GRAPHICS CONTROLLER FOR FPGA

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

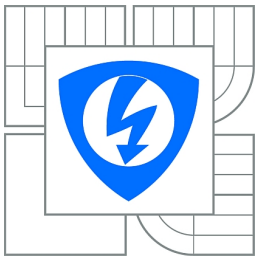
AUTOR PRÁCE — Bc. MAROŠ ROLKO

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARIÁN PRISTACH

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav mikroelektroniky

Diplomová práce

magisterský navazující studijní obor
Mikroelektronika

Student:	Bc. Maroš Rolko	ID:	115268
Ročník:	2	Akademický rok:	2013/2014

NÁZEV TÉMATU:

Grafický kontrolér pro obvody FPGA

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s dostupnými standardy pro grafické karty a s možnostmi implementace grafického kontroléru do obvodu FPGA. Na základě získaných informací navrhněte architekturu kontroléru s možností připojení k mikroprocesoru s operačním systémem Linux. Navržený grafický kontrolér popište v jazyce VHDL a implementujte do obvodu FPGA Spartan-3. Kontrolér bude podporovat vykreslování základních geometrických objektů a textu na základě příkazů z nadřízeného systému.
Součástí práce bude i vzorový příklad s procesorem, který bude generovat data pro vykreslování testovacích obrazců.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání:	10.2.2014	Termín odevzdání:	29.5.2014

Vedoucí práce: Ing. Marián Pristach
Konzultanti diplomové práce:

prof. Ing. Vladislav Musil, CSc.
Předseda oborové rady

Abstrakt

Táto diplomová práca sa zaoberá návrhom 2D grafického kontroléra pre obvody FPGA. Skladá sa z dvoch častí. V prvej fáze sa venuje rozboru 2D akcelerácie a rozhrania operačného systému Linux pre komunikáciu so zobrazovacími zariadeniami. Druhá časť obsahuje samotný návrh grafického kontroléra a jeho funkčnú implementáciu. Súčasťou práce je popis jednotlivých komponentov z ktorých sa kontrolér skladá a vyhodnotenie parametrov výslednej implementácie. Pre testovanie na konkrétom FPGA obvode je vytvorené testovacie zapojenie, pridávajúce podporu pre použité periférie a vytváranie testovacích dát.

Kľúčové slová

VHDL, FPGA, grafický kontrolér, Linux, 2D akcelerácia, VGA

Abstract

This diploma thesis is about design of a 2D graphics controller for FPGA circuits. It consists of two parts. In the first phase, it analyses 2D acceleration and interface for communication with display devices of the operation system Linux. The second part contains design of graphics controller itself and its implementation. Part of the thesis is description of components that the controller consists of and evaluation of resultant implementation. For testing purposes on selected FPGA circuit, test modules adding support for used peripherals and test data generation are created.

Key words

VHDL, FPGA, graphics controller, Linux, 2D acceleration, VGA

Bibliografická citácia mojej práce:
ROLKO, M. *Grafický kontrolér pro obvody FPGA*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačných technológií, 2012. 67 s. Vedúci semestrálnej práce Ing. Marián Pristach.

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technicka 12, CZ-61600 Brno, Czech
Republic
http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Prehlásenie

Prehlasujem že som túto vysokoškolskú kvalifikačnú prácu na tému *Grafický kontrolér pro obvody FPGA* som vypracoval samostatne pod vedením vedúceho diplomovej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení časti druhej, hlavy VI. diel 4 Trestného zákonníka č 40/2009Sb.

V Brne dňa 28.5. 2014 .....

podpis autora

Poďakovanie

Ďakujem vedúcemu diplomovej práce projektu Ing. Mariánovi Pristachovi za účinnú metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady pri spracovaní mojej diplomovej práce.

V Brne dňa 28.5. 2014 .....

podpis autora

Obsah

Úvod

Cieľom práce je návrh grafického kontroléra generujúceho výstupný signál pre monitor cez VGA port. Návrh bude následne implementovaný do hradlového poľa Spartan 3. Zadanie je možné rozložiť na tri časti. Prvá časť obsahuje teoretický rozbor zaužívaných princípov v oblasti 2D akcelerácie a možnosti operačného systému Linux pri komunikácií so zobrazovacími zariadeniami.

Druhá časť obsahuje principiálny návrh grafického kontroléra a popis blokov potrebných pre jeho správne fungovanie. Rozpísané sú algoritmy pre vykonávanie jednotlivých funkcií a podmienky ktoré musia byť zahrnuté v implementácií. Navrhnutý systém sa skladá z kontrolérov komunikačných rozhraní, akceleračných jednotiek, riadiacej logiky a medzipamätí. Pre otestovanie funkcie obvodu je nutné vytvoriť dodatočné bloky na zabezpečenie kompatibility s vývojovou doskou.

Záverečná časť obsahuje popis implementácie navrhnutého grafického zariadenia. Kontrolér tvorený z modulov popísaných na úrovni RTL. Modulárnosť systému umožňuje jednoduché rozširovanie a adaptáciu na iný hardvér.

Požadované parametre obvodu sú minimálna pracovná frekvencia 50 MHz, podpora pre zobrazovací mód 640×480 s obnovovacou frekvenciou 60 HZ, lineárna zobrazovacia pamäť a implementovanie akceleračných funkcií pre zobrazenie jednoduchých geometrických útvarov a znakov. Grafický kontrolér interne pracuje s farebnou hĺbkou 24 bitov, ktorá je v testovacom zapojení limitovaná na 3 bity, vzhľadom na možnosti použitej vývojovej dosky. Súčasťou zadania je rozbor rozhraní pre pripojenie navrhnutého grafického kontroléru poskytovaných operačným systémom Linux.

1 Popis použitých nástrojov

1.1 Digitálne obvody

Digitálne obvody sa primárne využívajú na tvorenie logických funkcií, aritmetické operácie a ukladanie dát. Jednoduché logické funkcie je možné vytvárať pomocou diskrétnych súčiastok, ale tento prístup má značné nevýhody a je pri vyššej komplexnosti nepoužiteľný. Ďalším typom digitálnych obvodov sú mikrokontroléry, umožňujúce programovanie pomocou aplikačnej pamäte a nemennej inštrukčnej sady. Pri niektorých operáciách je nevyhnutná rozsiahla paralelizácia výpočtov a ich softvérová implementácia pomocou obmedzených inštrukcií je neefektívna. Na realizáciu špecifických digitálnych funkcií je možné použiť zákaznícky integrovaný obvod ASIC (Application Specific Integrated Circuit) čipy, ktoré sú navrhnuté výhradne pre zvolenú aplikáciu. Vyznačujú sa vysokou optimalizáciou a efektivitou ale aj nákladným vývojom. Najflexibilnejšie riešenia sú hradlové polia FPGA (Field Programmable Gate Array), založené na statickej pamäti SRAM (Static Random Access Memory) a konfigurovateľných logických blokoch alebo v prípade menších návrhov programovateľné logické čipy CPLD (Complex Programmable Logic Device) využívajúce polia hradiel typu and alebo or. Tieto obvody poskytujú programovateľnú logiku, ktorú je možné meniť podľa požiadaviek aplikácie. [1]

1.2 Obvody FPGA

Obvod FPGA je konfigurovateľný integrovaný obvod, ktorý je možné programovať pomocou jazykov na popis hardvéru (HDL - Hardware Description Language) jazykov aj po výrobe. obvody FPGA nemajú preddefinovanú funkciu, preto je možné ich funkciu v prípade potreby upraviť alebo kompletne zmeniť bez fyzických zásahov do čipu. Prvé FPGA boli uvedené na trh v roku 1985 firmou Xilinx. Tieto čipy sa výrazne odlišovali od súčasných FPGA vytváraných procesmi s rozlíšením menším ako 30 nm a obsahujúcich desiatky miliónov hradiel.

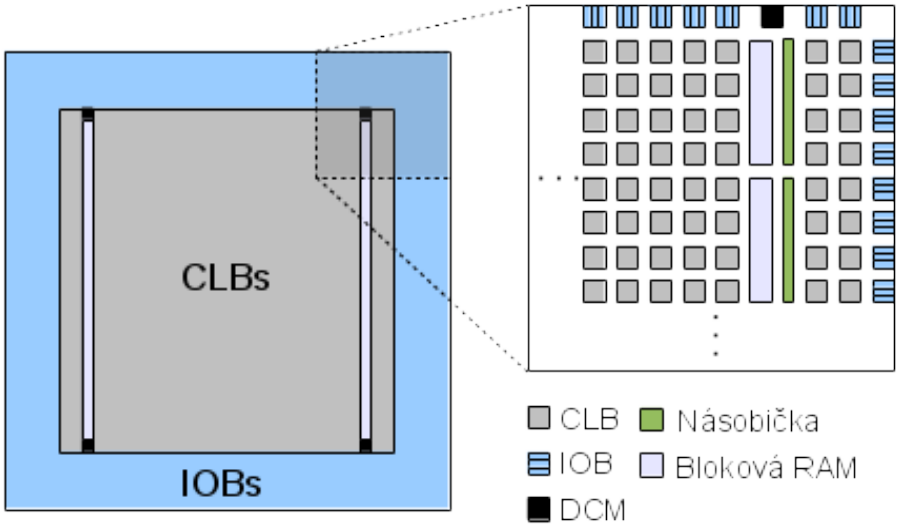
Základné bloky využívané pre kombinačnú aj sekvenčnú logiku sa nazývajú konfigurovateľné logické bloky. Tie sa potom skladajú z tzv. rezov (slices) ktoré obsahujú 2 náhľadové tabuľky - obvody pre generovanie logických funkcií, dva pamäťové prvky, realizované pomocou klopných obvodov typu D a dodatočnú logiku pre základnú aritmetiku a multiplexovanie. Konfigurovateľné logické bloky môžu byť pripojené ku ďalším funkčným blokom. Tie umožňujú vykonávať vybrané operácie so značne vyššou efektivitou ako ich ekvivalenty realizované pomocou štandardných buniek. Pripojenie FPGA ku zvyšku systému je prebieha s využitím vstupno-výstupných blokov .[2]

1.3 Spartan 3 Starter Kit Board

Spartan 3 Starter Kit Board je vývojová doska slúžiaca ako vývojová platforma pre FPGA Spartan 3 od firmy Xilinx. Hlavným komponentom dosky je FPGA čip od firmy Xilinx, Spartan 3 XC3200, pamäť sa skladá z 2 Mbit programovateľnej pamäti Flash PROM (Programmable Read Only Memory) a asynchrónnej SRAM s kapacitou 1 MB. Na komunikáciu s perifériami používa Spartan 3 Starter Kit sériový port, VGA a PS/2 porty, programovanie FPGA a PROM prebieha cez rozhranie JTAG (Joint Test Action Group). Pre vývojové účely sa na doske nachádzajú nasledujúce komponenty: kryštálový oscilátor generujúci signál s frekvenciou 50 MHz, štvormiestny sedem-segmentový displej, výstupné LED (Light Emmiting Diode), tlačidlá a prepínače. Spartan 3 starter board je možné rozšíriť pomocou externých modulov. [3]

1.4 Xilinx Spartan-3 XC3S200 FPGA

Spartan 3 vyvinula firma Xilinx v roku 2003. Okrem konfigurovateľných logických blokov a vstupno-výstupných blokov obsahuje ďalšie 3 typy logických obvodov a to 216 kbit blokovú pamäť RAM (Random Access Memory), dvoj-vstupovú 18-bitovú násobičku a obvod pre manažment digitálneho hodinového signálu [4]. Architektúra čipu Spartan 3 je zobrazená na obr. 1.1.



Obr. 1.1: Architektúra Spartan 3 FPGA. [4]

V práci bude použitý model XC3S200 v 256 pinovom BGA (Ball Grid Array) puzdre (FT256), programovaný cez rozhranie JTAG pomocou HDL jazykov, napríklad VHDL (VHSIC Hardware Description Language) alebo Verilog. Frekvencia použitého hodinového signálu je 50 MHz.

Parametre definujúce možnosti použitého čipu sú v tab. 1.1.

Tab. 1.1: Parametre čipu Spartan 3 XC3S200. [4].

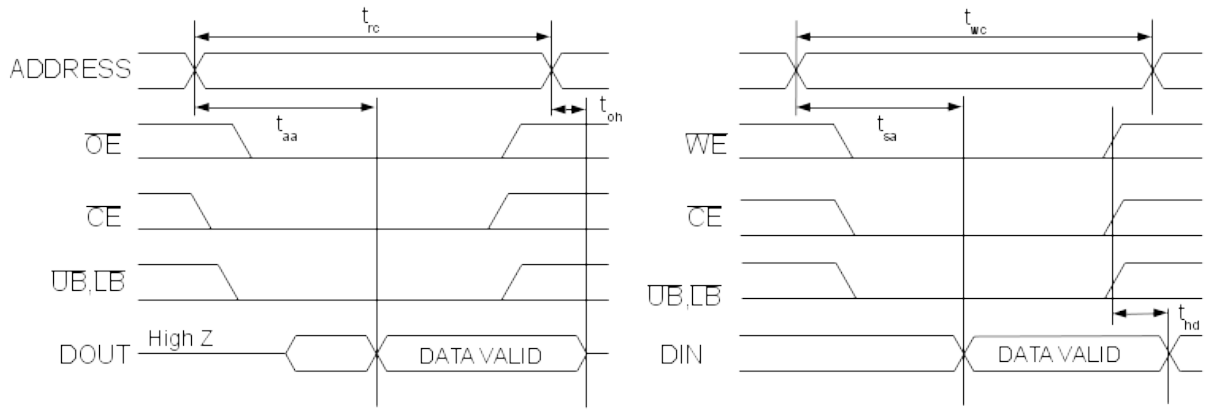
Ekvivalentné hradlá	200000
4-vstupové LUT	3840
Klopné obvody D	1920
Bloková ram	216Kib
Násobičky	12
Vstupy/výstupy	173
Diferenčné I/O páry	76
Napäťový rozsah I/O pinov	1,2V-3,3V

1.5 Asynchrónna SRAM

Vývojová doska obsahuje dva pamäťové moduly ISSI IS61LV25616AL. Veľkosť jednotlivých pamätí je 256K×16 bitov, celková kapacita dostupná pre FPGA je 1 MB. Jedná sa o pamäťové čipy vytvorené pomocou technológie CMOS (Complementary Metal Oxid Semiconductor) fungujúce asynchrónne bez hodinového signálu. Tieto moduly sú ovládané niekoľkými signálmi: *ce* ktorý aktivuje zvolený čip, *we* kontrolujúci zápis do pamäte, *oe* ovládajúci čítanie z pamäte a signály pre sprístupnenie horného a dolného bajtu pamäťového bloku *ub*, *lb*. Všetky tieto signály sú aktivované logickou 0. Čipy používajú 18 bitové adresovanie a presun dát je realizovaný 16- bitovým troj-stavovým rozhraním. Pamäťové moduly použité vo vývojovej doske Spartan 3 starter board zdieľajú *we*, *oe* a adresové signály, ostatné sú špecifické pre každý modul. Prístupový čas pamäte je menej ako 10 ns, časovanie zápisu a čítania je zapísané v tab. 1.2 a zobrazené v obr 1.2. [5]

Tab. 1.2: Časové parametre SRAM. [5].

Dĺžka čítacieho cyklu	t_{rc}	10 ns	Dĺžka zapisovacieho cyklu	t_{wc}	10 ns
Prístup do pmäte	t_{aa}	< 10 ns	Nastavenie adresy	t_{as}	> 0 ns
Podržanie výstupu po čítaní	t_{oh}	> 2 ns	Podržanie dát po zápise	t_{hd}	> 0 ns



Obr. 1.2: Časovanie riadiacich signálov SRAM pri čítaní a zápise. [5]

1.6 VGA port

VGA rozhranie slúži na posielanie dát do zobrazovacích zariadení ako sú monitory. Odosielané sú synchronizačné signály určujúce rozlíšenie a obnovovaciu frekvenciu posielaného obrazu a tiež samotné informácie o jednotlivých pixeloch. Použitá vývojová doska Spartan 3 starter board využíva DB-15 konektor, čiže je priamo prepojiteľná s výstupnými zariadeniami pomocou štandardného VGA kábla. [3] Vzhľadom na absenciu RAMDAC (Random Acces Memory Digital-to-Analog Converter) modulov, slúžiacich na konverziu vektorov digitálneho signálu na analógové napäťové úrovne reprezentujúce intenzitu farby, je bitový rozsah VGA portu na doske obmedzený na 3 bity. Signály použité pre komunikáciu FPGA cez VGA sú nasledovné: VS pre vertikálnu synchronizáciu, HS pre horizontálnu synchronizáciu, R,G,B, ktoré určujú farbu zobrazenú na monitore, pričom R (red) je červená, G (green) je zelená a B (blue) modrá. Kombinovaním týchto signálov je možné vytvoriť celkovo 8 farieb. Farebné kombinácie sú v tab. 1.3.

Tab. 1.3: Zoznam farieb pre 3-bitový RGB vektor. [3]

R(červená)	G(zelená)	B(modrá)	Farba
1	1	1	biela
1	1	0	žltá
1	0	1	magenta
1	0	0	červená
0	1	1	zelenomodrá
0	1	0	zelená
0	0	1	modrá
0	0	0	čierna

Časovanie synchronizačných signálov je definované vo VESA štandarde, závisí na rozlíšení monitora a obnovovacej frekvencii. Signály VS a HS sú navzájom nezávislé.

1.7 Zbernica CLB

Zbernica CLB (Codasip Local Bus) primárne slúži na komunikáciu procesorového jadra s pamäťovými zariadeniami. Jedná sa o synchrónnu zbernicu s resetom aktívnym pri logickej jednotke. Skladá sa z nadradeného systému, ktorý odosiela požiadavky na jeden alebo viac podriadených modulov, ktoré požiadavky spracúvajú a odpovedajú na ne. Na komunikáciu sú použité dva 32-bitové signály pre príjem a zasielanie dát, adresný signál s variabilnou dĺžkou a kontrolné signály. Nadradený systém využíva nasledovné signály: *cs* – vybratie čipu, *we* – ovládanie zápisu, *sc* – počet blokov v 32 bitovom slove, *si* – index požadovaného bloku. Na odpoveď slúžia tri troj-bitové signály *aack* – prijatie požiadavky, *wresp* – ukončenie zápisu a *rresp* – ukončenie čítania. Tieto signály môžu nadobúdať nasledovné hodnoty: ACK (0b000) – potvrdenie požiadavky ukončenie transferu, WAIT (0b001) – transfer prebieha, UNALIGNED (0b100) – nesprávne zarovnanie adresy, požiadavka zamietnutá, OOR (0b101) - vstupná adresa je mimo adresný rozsah zariadenia. Ostatné slová sú rezervované a indikujú chybový stav. Rýchlosť prenosu je určená frekvenciou hodinového signálu *clk*. Na reset zbernice slúži signál *rst*. [6]

1.8 Xilinx ISE

Jedná sa o komplexné integrované vývojové prostredie (ISE – integrated software enviroment), určené na vývoj pre FPGA obvody od firmy Xilinx. Prostredie má v sebe zabudované množstvo nástrojov, určených pre rôzne fázy vývoja. Medzi hlavné patria syntetizátor HDL kódu (XST), nástroje na mapovanie (Map), rozmiestnenie a prepojenie komponent (Place&Route), softvér pre programovanie FPGA (Impact), simulátor ISIM. [7] Prostredie je vyvíjané od roku 1995. Podporované platformy sú Windows a Linux, v oboch prípadoch architektúry x86 a x86_64. Verzia použitá v tejto práci je 14.1. webpack pre Linux z roku 2012. Parametre syntézy, mapovania, rozmiestňovania a prepájania komponentov sú nastavené na štandardné hodnoty.

1.9 ModelSim

ModelSim je simulačný program vyvíjaný firmou Mentor Graphics, slúžiaci na simuláciu a verifikáciu HDL návrhov. Podporované jazyky sú VHDL, Verilog a s určitými obmedzeniami SystemC a SystemVerilog. [8] ModelSim je zároveň jadro verifikačného prostredia Questa, ktoré pridáva funkcie pre komplexné návrhy obvodov ASIC. Použitá verzia je 6.5f pre Linux.

2 2D akcelerácia

2.1 Konzorcium VESA

Za väčšinou štandardov v oblasti zobrazovacích technológií stoji konzorcium VESA (Video Electronics Standards Asociation). V roku 1988 bolo založené s cieľom vytvoriť jednotný štandard pre SVGA (super VGA) rozlíšenia 800×600 bodov. Po založení malo konzorcium deväť členov, v súčasnosti má členstvo viac ako 200 spoločností, vrátane hlavných výrobcov grafických akcelerátorov Intel, Nvidia, AMD a Qualcomm.

Medzi najznámejšie štandardy vydané VESA konzorciom partia VBE (VESA BIOS Extensions) – štandard definujúci základný spôsob komunikácie a identifikáciu grafických čipov, DisplayPort – rozhranie pre zobrazovacie zariadenia a E-EDID (Enhanced Extended Display Identification Data) zjednocujúci spôsob komunikácie pri zisťovaní podporovaných funkcií a stavu monitoru. [9]

Súčasťou VBE štandardu sú údaje o časovaní synchronizačných signálov pre rôzne rozlíšenia, vrátane rozlíšenia 640×480 s obnovovacou frekvenciou 60 Hz implementovaného v práci.

Na základe štandardov konzorcia vznikol univerzálny VESA ovládač pre X.Org Server, grafický server používaný v mnohých otvorených operačných systémoch, napríklad GNU/Linux alebo FreeBSD.

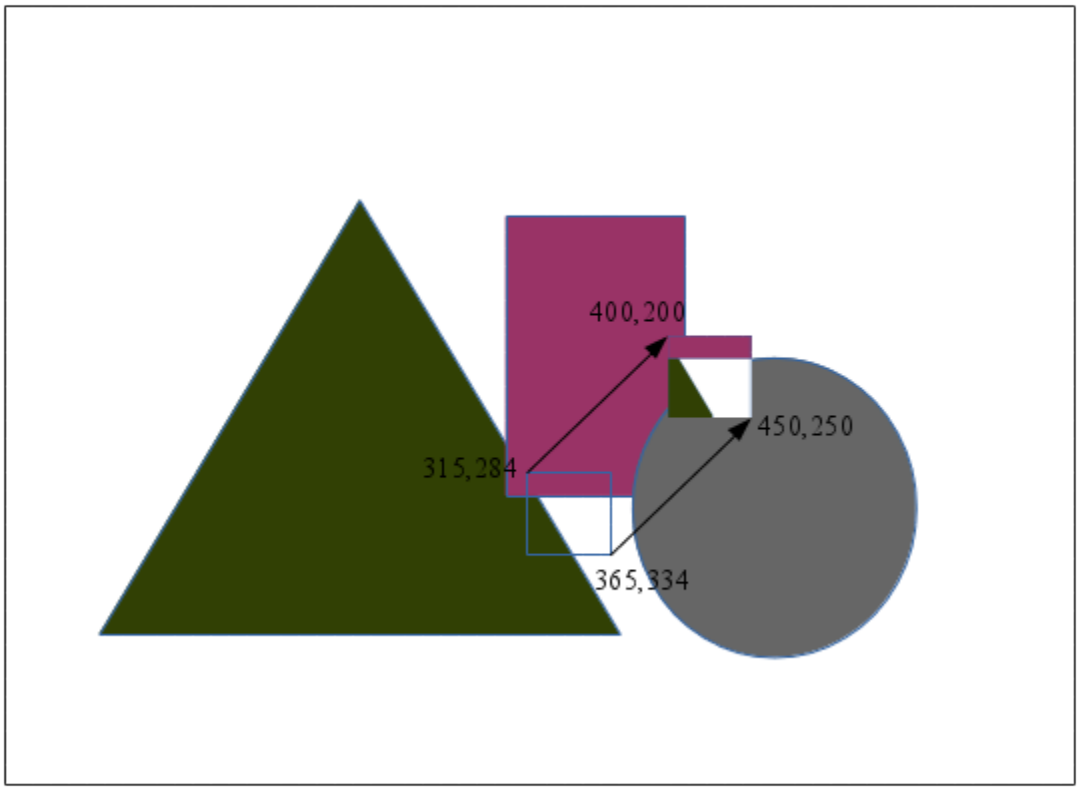
2.2 Základné princípy v 2D akcelerácii

Grafický akcelerátor má tri základné funkcie: komunikáciu s okolitými zariadeniami cez podporované rozhrania, poskytovanie pamäte, v ktorej je uložený zobrazovaný obraz a redukuje záťaž hlavného procesora samostatným vykonávaním výpočtovo náročných funkcií pri vykresľovaní obrazu.

Súčasné grafické akcelerátory podporujú širokú škálu akceleračných funkcií z oblasti 2D aj 3D vykresľovania ale aj univerzálne paralelné výpočty. Väčšina funkcií pre 2D akceleráciu spočíva primárne v práci s pamäťou, kopírovanie medzi rôznymi pamäťami systému alebo blokmi v rámci jednej pamäte. To zahŕňa napríklad BitBLT (Bit-boundary Block Transfer) operácie, zapisovanie bitových máp alebo vykresľovanie textových znakov.

BitBLT operácie slúžia na kopírovanie dát medzi jednotlivými časťami pamäte alebo rôznymi pamäťami. Buď je veľkosť kopírovaného bloku pevne stanovená, najčastejšie štvorec s rozmermi 8×8 pixelov, alebo variabilná, pri ktorej je zvolená oblasť obdĺžnik s nastaviteľnými rozmermi. Ako vstupné parametre sú súradnice prvého bodu referenčného

obdĺžnika, smer kopírovania a cieľová adresa zapisovaných dát. Niekedy miesto smeru a veľkosti kopírovanej oblasti môže byť zadaná poloha jej posledného pixelu, na základe ktorej je možné získať potrebné údaje. Smer čítania a zápisu je potrebné určiť, aby sa predišlo problémom s vykresľovaním. BitBLT operácie sú často využívané hardvérovo akcelerovanými grafickými prostrediami, súčasných operačných systémov. Kopírovanie štvorcovej oblasti zobrazovacej pamäti o veľkosti 50×50 bodov je zobrazené na obr. 2.1.



Obr. 2.1: BitBLT kopírovanie štvorca s rozmermi 50×50 pixelov.

Pri zapisovaní bitových máp sú dáta uložené do medzipamäte, aby mohli byť opakovane zapísané do hlavnej zobrazovacej pamäte na rôzne adresy. Využíva sa napríklad na urýchlenie zobrazovania kurzorov. [10]

Akcelerácia fontov funguje na podobnom princípe. Dáta o jednotlivých symboloch sú buď dynamicky načítané do určenej časti pamäte z nadradeného procesoru počas zavádzania systému, alebo uložené vo vnútornej ROM pamäti. Odtiaľ sú následne na základe príkazov centrálnej riadiacej jednotky kopírované do hlavnej zobrazovacej pamäti.

Ďalšia kategória funkcií dvoj-dimenzionálnej grafickej akcelerácie je vykresľovanie jednoduchých geometrických útvarov. Najbežnejšie objekty sú trojuholníky, obdĺžniky a čiary. Hardvérové vykresľovanie trojuholníkov je používané najčastejšie pri zobrazovaní 3D objektov, zložené z potencionálne veľkého množstva malých trojuholníkov. V 2D prostredí sa trojuholníky takmer nevyužívajú. Primárne používané objekty sú čiary,

vykresľované pomocou rôznych algoritmov. Prvým bol Bresenhamov algoritmus, ktorý sa stal populárny pre jeho nízku výpočtovú náročnosť a používanie iba jednoduchých aritmetických operácií bez potreby desatinných čísiel. V súčasnosti existujú značne pokročilejšie algoritmy, obsahujúce podporu antialiasingu a priehľadnosti.

Akcelerované vykresľovanie iných a komplexnejších útvarov nebýva štandardne zabudované v zobrazovacích zariadeniach, vzhľadom na ich nízku frekvenciu používania.

Niektoré čipy majú pridanú akceleráciu ďalších operácií, napríklad korekciu farieb obrazu, aplikovanie rôznych transformačných filtrov, priehľadnosť a spracovanie videa. Tieto rozširujúce funkcie sú často špecifické pre konkrétny hardvér a vyžadujú podporu zo strany ovládačov a softvéru.

Moderné grafické prostredia využívajú funkcie zahrnuté v 3D štandardoch, pretože súčasné grafické čipy majú unifikovanú výpočtovú architektúru a 2D funkcie sú emulované hardvérom určeným na 3D akceleráciu.

2.3 Existujúce štandardy v oblasti 2D akcelerácie

Prvým štandardom v oblasti zobrazovacích čipov je VGA štandard z roku 1992 od firmy IBM. Popisuje jednotlivé funkcie a komponenty zobrazovacích kontrolérov. Ďalej definuje podporované rozlíšenia pre grafický mód aj textový mód. Maximálne rozlíšenie definované v štandarde je 640×480@70 Hz s farebným rozsahom 16 farieb. 8 bitová farebná hĺbka je možná iba pri rozlíšení 320×240 a obnovovacej frekvencií 70 Hz. Pamäť zariadenia je rozdelená do 4 blokov o veľkosti 64 KB. Štandard má značné nedostatky, vyplývajúce z obmedzení hardvéru a potrieb danej doby. Hlavným nedostatkom je možnosť pristupovať vždy iba k 1 bloku pamäti a nelineárne rozloženie zobrazovacej pamäte. Okrem toho sa podporované zobrazovacie módy rýchlo zastarali. Štandard vyžaduje VGA BIOS aj PC BIOS, je viazaný na architektúru x86 a v súčasnosti značne zastaraný. [11]

Rozšírenia pre VGA štandard sú označované ako SVGA. Tieto rozšírenia uvádzajú nové, pokročilejšie zobrazovacie módy, rozšírenie zobrazovacej pamäte a ďalšie funkcie, napríklad akceleráciu. SVGA rozhranie nikdy nebolo štandardizované, preto sa jeho implementácie, podporované vlastnosti a komunikačné rozhrania značne líšia medzi jednotlivými čipmi. Z toho vyplynula absencia univerzálnych ovládačov a náročnosť podpory zo strany softvéru využívajúceho tieto rozšírené funkcie. Ako čiastočné riešenie spomenutých problémov s nekompatibilitou SVGA zariadení, vznikol štandard VBE. Tento štandard zjednocuje identifikáciu zariadení, hlásenie podporovaných funkcií a zobrazovacích módov. [12]

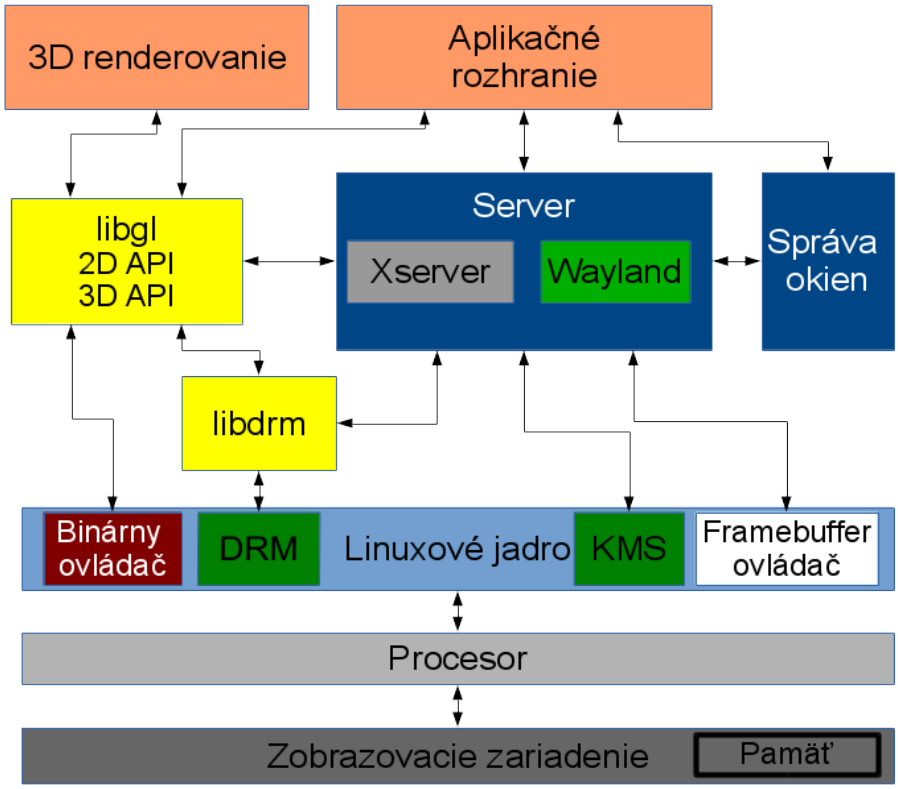
Štandard VBE sa postupne rozširoval a verzia 2.0 priniesla v prvom rade podporu lineárnej zobrazovacej pamäte. To znamená že zobrazovacia pamäť sa skladá z jedného súvislého bloku. Súčasná verzia štandardu je 3.0 a existujú jeho rozšírenia definujúce

akceleračné funkcie a komunikačné rozhrania. Štandard je viazaný na x86 architektúru, využíva interné registre procesora. Rovnako je na jeho aplikáciu potrebný systémový a VGA BIOS. [12]

V oblasti embbeded zariadení a soft-procesorov, pre ktoré je grafický kontrolér navrhovaný, neexistuje zobrazovací štandard, podobný VBE. Implementácia miesto toho využíva linuxové API (Application Programming Interface) a špecifický ovládač.

Čiastočne do oblasti 2D akcelerácie zasahujú štandardy OpenGL a OpenGL ES. Obidva štandardy sú však skôr zamerané na 3D akceleráciu a zahŕňajú funkcie mimo rozsah zadania. Funkcie definované v rámci týchto štandardov nie sú implementované. Rovnako komponent pre 2D akceleráciu DirectDraw zo známeho multimediálneho API DirectX od spoločnosti Microsoft nie je podporovaný. Čiastočnú podporu týchto štandardov je možné dodatočne doplniť pomocou ovládačov.

2.4 Linuxové rozhranie pre zobrazovacie zariadenia



Obr. 2.2: Štruktúra grafických rozhraní Linuxu. [13]

Operačný systém GNU/Linux a jeho grafické nadstavby poskytujú široké spektrum rôznych API, generických ovládačov a podporu pre väčšinu súčasných štandardov. Ich štruktúra je zobrazená na obr. 2.2.

Na úrovni jadra existujú dve rozhrania poskytujúce API pre ovládače, hardvér

a užívateľské prostredie. Primárny je modul jadra DRM (Direct Rendering Manager). Tento modul poskytuje rozhranie pre hardvérové ovládače, knižnice a používateľské programy využívajúce hardvérovú akceleráciu.

Pre jednoduché špecifické systémy a starší hardvér existuje Frambuffer API, sada funkcií a doporučení pre tvorbu ovládačov. Toto rozhranie je značne obmedzené, podporuje iba základnú prácu s pamäťou zobrazovacieho zariadenia a malú sadu funkcií pre 2D akceleráciu.

Existuje niekoľko univerzálnych ovládačov pre zobrazovacie zariadenia, avšak všetky sú viazané na konkrétne platformy, pre ktoré nie je navrhovaný grafický čip cielený, najčastejšie x86 procesory s podporou VBE štandardu.

Funkcie pre grafické prostredia sú poskytované grafickými servermi, najpoužívanejší je Xserver využívajúci protokol X11. V súčasnosti je už značne zastaraný a intenzívne sa pracuje na nástupcovi Wayland server. Tieto programy využívajú funkcie poskytované jadrom a ovládačmi jadra, navyše vyžadujú vlastné ovládače. V prípade že tie neexistujú, je použitý generický VESA ovládač, obsahujúci iba základné zobrazovacie funkcie. [13]

2.5 Tvorba Linuxového framebuffer ovládača

Frambuffer API pre písanie ovládačov zobrazovacích zariadení má povinnú preddefinovanú formu. Ovládače sú silne závislé na štruktúrach deklarovaných a popísaných v hlavičkovom súbore fb.h. Jedná sa o štruktúry *fb_var_screeninfo*, *fb_fix_screeninfo*, *fb_monospecs* a *fb_info*.

Prvá spomenutá, *fb_var_screeninfo*, obsahuje popis funkcií, ktoré môžu byť nastavené jadrom a zmenené počas normálneho fungovania grafického čipu. Druhá, *fb_fix_screeninfo* obsahuje zoznam vlastností karty, ktoré sú nastavené pri nastavení zobrazovacieho módu karty a nie je možné ich meniť v rámci daného módu. Štruktúra *fb_monospecs* slúži na ochranu monitoru. Posledná spomenutá, *fb_info* definuje momentálny stav zariadenia.

Ovládač musí povinne obsahovať inicializačné funkcie, funkcie pre prácu s pamäťou, funkcie pre nastavovanie vlastností zobrazovacieho zariadenia, voliteľne akceleračné funkcie, základnú správu napájania a dodatočné vlastnosti.

Na inicializáciu sú definované funkcie *xxfb_init* a *xxfb_setup,* obe vracajúce hodnotu typu integer. Funkcia *xxfb_init* nemá žiadny vstupný parameter a nastavuje počiatočný stav grafického čipu počas zavádzania systému. Rovnako je možné pri spustení systému použiť *xxfb_setup,* ktorá inicializuje kartu na základe parametrov zadaných v zavádzači.

Ďalej je povinné deklarovať funkcie z fb_ops časti štruktúry fb_info. Základné sú *xxfb_open* a *xxfb_release* slúžiace na získanie prístupu do pamäte zariadenia. Tieto funkcie môžu byť používané modulmi fbcon (Framebuffer console) alebo fbdev(Framebuffer

device). Na nastavovanie parametrov grafického čipu je použitá funkcia *xxfb_set_var*. V prípade zariadení používajúcich nelineárnu zobrazovaciu pamäť, čiže adresný priestor pamäte nie je súvislý blok, je nevyhnutné pre korektnú prácu s pamäťou definovať metódy IO operácií *xxfb_write* a *xxfb_read*, slúžiace na operácie zápisu a čítania. Ďalší krok pri tvorení ovládača je vytvorenie funkcie *xxfb_set_colreg*, ktorá sa stará o preklad medzi farebnými mapami podporovanými jadrom a vnútorným kódovaním farieb zobrazovacieho zariadenia. Mapovanie zobrazovacej pamäte je riešené pomocou funkcie *xxfb_mmap*.

Ostatné funkcie sú nepovinné a v prípade absencie podpory v hardvéri sú nahradené hodnotou NULL a riešené softvérovými prostriedkami. Funkcia *xxfb_blank* slúži na základnú správu napájania, nastavuje uspatie, vypnutie alebo pozastavenie funkcie zobrazovania v karte počas dlhodobejšej nečinnosti. Podpora pre ďalšie možnosti grafického čipu je pridávaná pomocou funkcií *xxfb_pan_display*, *xxfb_ioctl*, *xxfb_rotate* a ďalších.

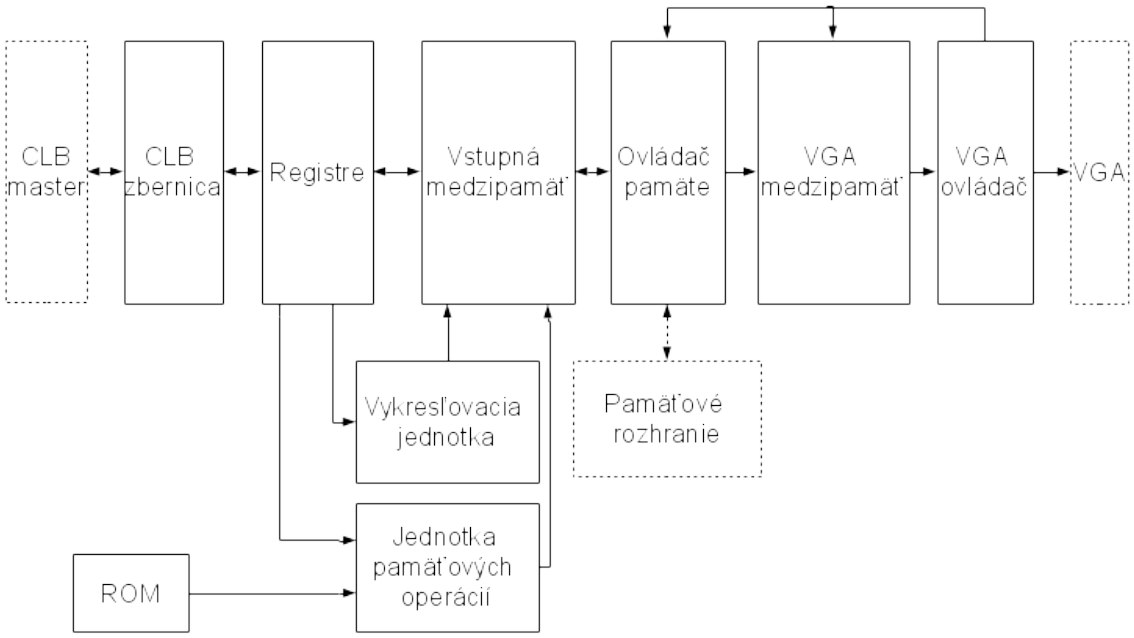
Hardvérová akcelerácia vyžaduje vytvorenie funkcií *xxfb_accel_init* – inicializácia akceleračných prostriedkov zariadenia a *xxfb_accel_done* – ukončenie používania akcelerácie. Základné podporované akceleračné funkcie sú *xxfb_fillrect* – vyplnenie obdĺžnikovej oblasti, *xxfb_imgblt* – kopírovanie obrázku zo systémovej pamäti do zobrazovacej a *xxfb_copyarea* – kopírovanie obdĺžnikových plôch v rámci zobrazovacej pamäti na základe zadaných adries. Ak neexistuje podpora pre tieto funkcie zabudovaná v zobrazovacom zariadení, Linuxové jadro poskytuje ich softvérový ekvivalent. [14]

Ovládač pre navrhnutý systém je ďalej zjednodušený, primárne pre redukované možnosti nastavenia grafického jadra. Jadro poskytuje iba jeden zobrazovací mód s pevne stanovenými parametrami ako rozlíšenie, farebná hĺbka a obnovovacia frekvencia. Inicializačné funkcie by tak boli pevne nastavené na konštantné hodnoty. Použitie lineárneho adresovania zobrazovacej pamäte navyše umožňuje použitie prednastavených funkcií na čítanie a zápis. Najkomplikovanejšie je pridanie podpory pre hardvérovú akceleráciu zabudovanú v navrhnutom grafickom obvode a zaistenie kompatibility medzi podporovanými akceleračnými funkciami operačného systému a skutočnými možnosťami systému. Správa napájania nie je implementovaná takže funkcia *xxfb_blank* nemusí byť definovaná v rámci ovládača. Rovnako väčšinu dodatočných vlastností je možné vynechať. Ovládač je zároveň silne závislý na použitej zbernici a jej ovládači, preto je nutné vziať v úvahu spôsob komunikácie a zápisu do registrov alebo zobrazovacej pamäte.

Vzhľadom na jednoduchosť a nezávislosť navrhovaného grafického akcelerátora na použitom procesore by mal byť ovládač do značnej miery multiplatformný.

3 Návrh systému

3.1 Architektúra grafického kontroléru



Obr. 3.1: Blokový návrh grafického kontroléru

Grafický kontrolér umožňuje zobrazovanie výstupných dát na monitor. Primárne slúži ako pamäťové zariadenie, do ktorého sa ukladá bitová mapa, ktorá je následne odosielaná na zobrazenie cez zvolené komunikačné rozhranie. Okrem toho grafické kontroléry môžu obsahovať ďalšie funkcie pre 2D a 3D akceleráciu.

Vstup prebieha cez zbernicu CLB priamo pripojenú ku nadradenému procesoru. Kontrolér má 19-bitový adresný priestor, prístup prebieha priamo do zobrazovacej pamäte zariadenia alebo do registrov s rezervovanými adresami. Rozsahy sú nasledovné: 0x00000-0x7FFEF operácie pre manipuláciu so zobrazovacou pamäťou, adresy 0x7FFF0-0x7FFFB ukazujú na rozšíriteľné registre pre špeciálne akceleračné funkcie. Výstupné signály sú kompatibilné s VGA synchronizačnými signálmi a 24 bitovou farebnou hĺbkou. Rozhranie pre komunikáciu s pamäťou obsahuje 24-bitovú zbernicu pre čítanie a zápis, 19 bitové adresovanie a trojicu riadiacich signálov.

Navrhnutý systém sa skladá z ôsmych vzájomne komunikujúcich modulov. Bloková schéma navrhnutého zariadenia je zobrazená na obr. 3.1. Príjem dát, vstupných príkazov z hlavného procesora a spätnú komunikáciu s hlavným procesorom zaobstaráva kontrolér zbernice CLB. V nasledujúcom module, registrovej časti, je podľa typu požiadavky vstup následne rozdelený do štyroch kategórií: zápis do pamäte, čítanie z pamäte, zápis do registrov a čítanie z registrov. Práca s registrami prebieha okamžite, v prípade potreby

zápisu do pamäte sú dáta odosielané do vstupnej medzipamäte typu FIFO pre zlepšenie výkonu systému. Vzhľadom na povahu použitej zbernice CLB nie je možné rovnako ukladať požiadavky na čítanie z pamäte, preto tie sú odosielané prioritne, aby sa minimalizovala doba čakania na požadované dáta.

O prácu s pamäťou sa stará pamäťový kontrolér. Jeho primárnou úlohou je zabrániť konfliktom pri prístupe do pamäte. Rozhranie umožňuje pripojiť rôzne typy pamätí, väčšina však vyžaduje externý ovládač na preklad komunikácie zo systému na konkrétne vstupné a výstupné signály konkrétnej pamäte.

Na základe synchronizačných signálov z VGA kontroléra, pamäťový kontrolér odosiela dáta jednotlivých pixelov do medzipamäte VGA typu FIFO, kde je pred samotným vykreslením uložený nasledujúci riadok. VGA kontrolér číta informácie z tejto pamäte a odosiela ich do výstupu. Okrem toho obsahuje aj dva čítače, starajúce sa o generovanie potrebných časovacích signálov. Taktovanie VGA kontroléra na frekvenciu 25 MHz prebieha pomocou pseudo-hodinového signálu vytvoreného frekvenčnou deličkou.

Pri návrhu bolo zvážené pridanie podpory štandardu VBE 2.0+, čo by umožnilo fungovanie grafického jadra s generickým ovládačom *fbdev* na platforme Linux, avšak implementácia by vyžadovala čiastočnú emuláciu x86 architektúry a dodatočnú logiku slúžiacu ako VGA BIOS. [12] Vzhľadom na neexistenciu štandardov zobrazovacích zariadení pre cieľový procesor, zvolený postup pre pridanie linuxovej podpory je vytvorenie vlastného framebuffer ovládača. Grafický kontrolér neobsahuje žiadne pokročilé funkcie, takže by nedokázal využiť výhody nových a komplexnejších rozhraní.

3.2 Komunikačné rozhrania kontroléru

Pre komunikáciu s nadradeným procesorom je použitá zbernica CLB. Grafické jadro využíva CLB slave mód, kde zariadenie odpovedá na požiadavky z nadradeného systému, ale samostatne neiniciuje žiadnu komunikáciu. Táto zbernica sa skladá zo kontrolných signálov z nadradeného systému *cs*, *we*, *si*, *sc*, adresy *a* o šírke 19 bitov, 32 bitového vektora odosielaných dát *d* a signálov odpovedí skladajúcich sa z troch trojbitových vektorov *wresp*, *rresp*, *aack* a vektora dát *q* prečítaných z pamäte o šírke 32 bitov. Podrobný popis komunikácie sa nachádza v kapitole 3.4. [6]

Komunikačné rozhranie medzi grafickým jadrom a radičom pamäťových čipov je neštandardné a prispôsobené potrebám aplikácie s prihliadnutím na minimalizáciu dopadu na zdroje cieľového FPGA obvodu. Odosielané a prijímané dáta majú rozsah 24 bitov, adresa je 19 bitová. Riadiace signály sú tri, požiadavka na čítanie, zápis a potvrdenie vykonania požiadavky, všetky majú veľkosť 1 bit. Transfer dát prebieha nasledovne: zvolená požiadavka je nastavená, spolu s dátami a adresou a kontrolér čaká na odpoveď ovládaču. V momente odpovede sú nastavené nové hodnoty, prípadne uložené prečítané dáta.

Tab. 3.1: Zoznam vstupných a výstupných portov grafického kontroléru

Názov	Typ	Funkcia	Veľkosť v bitoch
clk	vstup	hodinový signál	1
rst	vstup	asynchrónny reset	1
cs	vstup	CLB riadiaci signál	1
a	vstup	CLB vstupná adresa	19
si	vstup	CLB riadiaci signál	2
sc	vstup	CLB riadiaci signál	3
d	vstup	CLB vstupné dáta	32
aack	výstup	potvrdenie pre CLB master modul	3
rresp	výstup	potvrdenie ukončenia čítania	3
wresp	výstup	potvrdenie ukončenia zápisu	3
q	výstup	CLB výstupné dáta	32
memory_address	výstup	adresa pre pamäťový ovládač	19
memory_write_data	výstup	dáta určené na zápis	24
memory_write_req	výstup	požiadavka na zápis	1
memory_ready	vstup	potvrdenie ukončenia pamäťovej	1
memory_read_data	vstup	dáta prečítané z pamäte	24
memory_read_req	výstup	požiadavka na čítanie	1
hsync	výstup	horizontálna synchronizácia VGA	1
vsync	výstup	vertikálna synchronizácia VGA	1
rout	výstup	intenzita červenej farby	8
gout	výstup	intenzita zelenej farby	8
bout	výstup	intenzita modrej farby	8

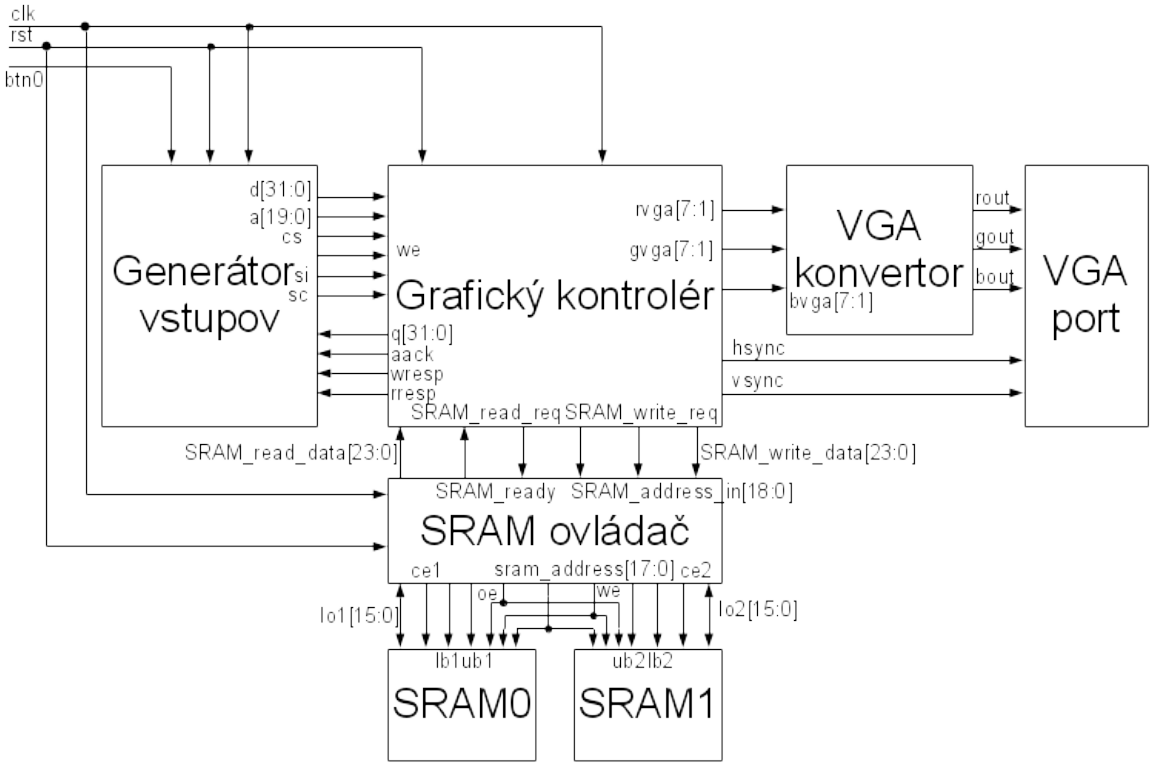
Ako výstup pre monitor je využité VGA rozhranie. Farebná hĺbka je pevne 24 bitov, v prípade, že je požadované iné nastavenie, je nutná externá konverzia. Signály *h_sync* a *v_sync* sú periodicky odosielané v súlade s cieľovým časovaním. Jednotlivé porty a ich základné informácie sú zobrazené v tab. 3.1.

3.3 Blokový návrh testovacieho zapojenia

Testovanie a implementácia do použitej vývojovej dosky Spartan 3 vyžadujú použitie dodatočných modulov. Jednoduché modelovanie vstupu zabezpečuje generátor vstupných dát, ktorý simuluje logiku CLB master modulu a odosiela dáta z ROM pamäte. Vzhľadom na obmedzenú kapacitu pamäte čipu je zobrazovacia pamäť zaplňovaná periodickými blokmi o veľkosti 16 pixelov. Na základe vstupného impulzu prebieha zapisovanie do pamäte až po adresu x4AFFF ktorá predstavuje posledný zobrazený bod na obrazovke pri rozlíšení 640×480. Tieto dáta sú následne spätne prečítané pre overenie korektnosti funkcie čítania. Testovacie zapojenie je zobrazené na obr. 3.2.

Na komunikáciu medzi grafickým jadrom a dvoma SRAM čipmi je použitý modul na komunikáciu so SRAM. Ten obsahuje logiku na obsluhu vstupno-výstupnej dátovej zbernice a preklad ovládacích signálov. Okrem toho sa stará o prepínanie medzi jednotlivými modulmi, aby sa zabránilo konfliktom na zdieľaných zberniciach. To umožňuje 19 bitový adresný rozsah pamäte namiesto 18 bitového. Rovnako, vzhľadom na limitovanú veľkosť pamäťových buniek konvertuje uložené dáta z 24-bitového rozsahu komunikačnej zbernice grafického kontroléra na 16-bitový rozsah pamäťových buniek SRAM pamäte.

Vzhľadom na neprítomnosť RAMDAC čipov vo VGA rozhraní vývojovej dosky, určených na prevod viac-bitového digitálneho signálu na analógové napäťové úrovne je nutná konverzia 24 bitového signálu obsahujúceho informáciu o farbe zobrazovaného bodu na 3 bitový. O to sa stará VGA konvertor, ktorý jednoducho vyberá vždy najvyššie položený bit zo vstupnej farby.



Obr. 3.2: Bloková schéma testovacieho zapojenia navrhnutého grafického kontroléru.

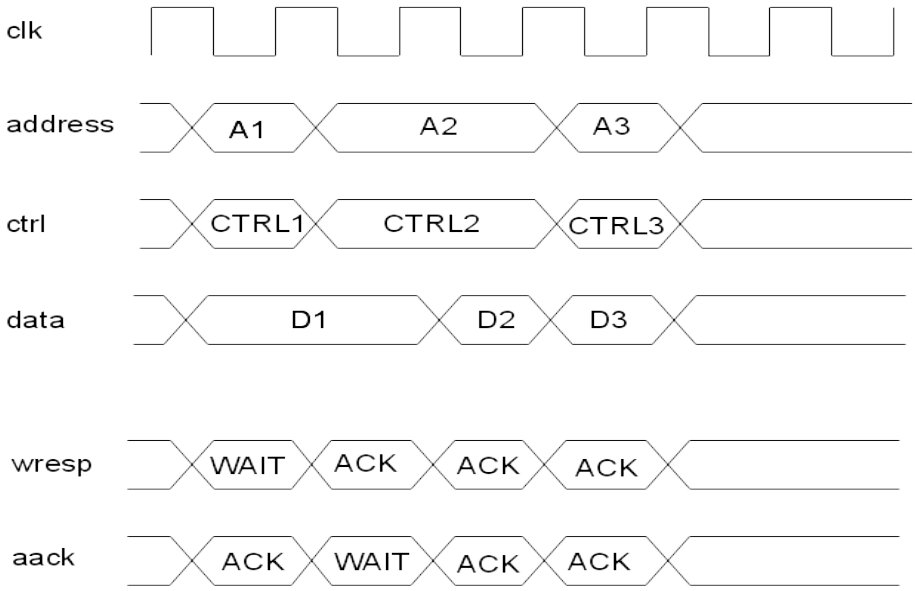
3.4 Rozhranie pre komunikáciu so zbernicou CLB

CLB slave modul slúži na komunikáciu kontroléra s nadradeným systémom cez zbernicu CLB. Požiadavka na kontrolér je iniciovaná pomocou signálu *cs*, typ požiadavky určuje signál *we,* ak je nastavený na logickú 0, jedná sa o čítanie, v opačnom prípade o zápis.

Signály *si* a *sc* sú ignorované, ich aplikácia by vyžadovala pridanie extra logiky, zvýšenie zložitosti softvérového ovládača a zarovnanie adresy na 8 bitové bloky. Ako efektívnejšie riešenie je použité prijatie celého 32 bitového slova a ignorovanie prvých 8 bitov zľava v prípade potreby.

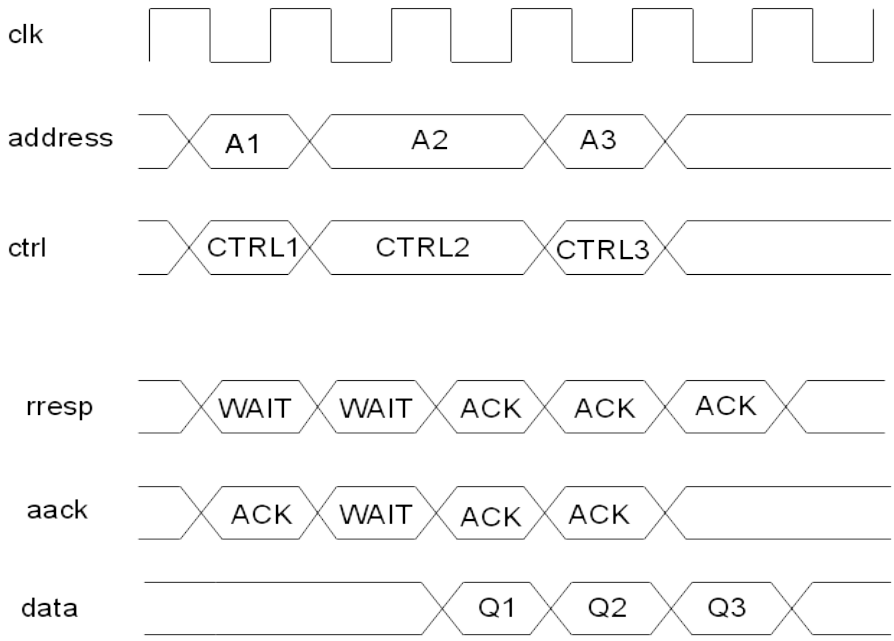
Prijatie a ukončenie požiadavky je riešené pomocou troch troj-bitových signálov. *aack* informuje o pripravenosti systému, *wresp* a *rresp* o ukončení zápisu, respektíve čítania. V prípade úspešného prijatia a dekódovania vstupných signálov je výstup *aack* nastavený na logickú úroveň 0b000 alebo ACK, v prípade čakania je odoslaná odpoveď WAIT reprezentovaná bitovým vektorom 0b001. Chyba je reprezentovaná sekvenciou 0b010.

Po ukončení zápisu je nastavený signál *wresp* na 0b000, čakanie a chyba zbernice sú rovnaké ako pri signále *aack*. Zápis je zobrazený na obr 3.3. Signál *rresp* je analogický k signálu *wresp*, ale je používaný počas čítania.



Obr. 3.3: Zapisovací cyklus s WAIT stavom. [6]

Pri čítaní je umožnené reťazenie, čo znamená že nové dáta a adresa sú nastavené a spracované ešte pred ukončením prechádzajúcej operácie. Čítanie s implementovaným reťazením a čakaním je zobrazené na obr. 3.4.



Obr. 3.4: Čítači cyklus s WAIT stavom. [6]

3.5 Registre

Jedná sa o centrálnu riadiacu jednotku grafického procesoru. Prijaté dáta sú spracované podľa zadanej adresy. Ak je vstup v rámci adresného rozsahu 0x00000-0xFFFEF, je požadovaný priamy prístup do zobrazovacej pamäte a dáta sú odoslané nasledujúcemu modulu bez ďalšieho spracovania. Zápis prebieha synchrónne, dáta sú odoslané do vstupnej medzipamäte v prípade, že nehlási stav limitného zaplnenia. Čítanie prebieha prioritne a asynchrónne, spracovávanie požiadaviek na zápis je pozastavené až do ukončenia operácie, z dôvodu minimalizácie čakania nadradeného systému.

Ostatné adresy z adresného priestoru sú rezervované pre registre akceleračných funkcií grafického kontroléra. Do týchto registrov sú zapisované vstupné parametre pre podporované operácie. Po nastavení zvoleného špecifického bitu sú dáta odoslané na spracovanie. Systém následne čaká na dokončenie operácie, aby sa predišlo problémom pri vykresľovaní nesynchronizovaných operácií pracujúcich s rovnakým umiestnením v pamäti. Popis adresného priestoru sa nachádza v tab. 3.2.

Tab. 3.2: Adresný priestor zariadenia

Adresa	Funkcia
0x00000 – 0x4AFFF	Funkčná oblasť zobrazovacej pamäte
0x4B000 – 0x7FFEF	Extra rozsah zobrazovacej pamäte
0x7FFF0 – 0x7FFF1	LINE register
0x7FFF2 – 0x7FFF3	ROM register
0x7FFF4 – 0x7FFF5	FILLRCT register
0x7FFF6 – 0x7FFF7	BRAMOP register
0x7FFF8 – 0x7FFF9	BRAMCPY register
0x7FFFA – 0x7FFFB	BITBLT register
0x7FFFC – 0x7FFFF	Nepoužíté registre

Dátový vektor pre priamy prístup do pamäte má rozsah 24 bitov, najvyšších 8 bitov signálu je nepoužitých. Registre využívajú plný 32 bitový rozsah vstupnej zbernice.

Tab. 3.3: Popis obsahu registrov pre špeciálne operácie

LINE	x7FFF0	X1	X2	BLUE	GREEN(3:0)			
	x7FFF1	GREEN(7:4)	RED	Y1	Y2	R	D	
ROM	x7FFF2	X	Y	ID	RESERVED			
	x7FFF3	BLUE	GREEN	RED	RESERVED	R	D	
FILLRCT	x7FFF4	X1	X2	BLUE	GREEN(3:0)			
	x7FFF5	GREEN(7:4)	RED	Y1	Y2	R	D	
BRAMOP	x7FFF6	BX	BY	MODE	RESERVED			
	x7FFF7	BLUE	GREEN	RED	M	RESERVED	R	D
BRAMCPY	x7FFF8	BX1	BY1	BX2	BY2	M	RESERVED	
	x7FFF9	FBX	FBY	RESERVED	R	D		
BITBLT	x7FFFA	X1	Y1	X2	RESERVED			
	x7FFFB	Y2	X3	Y3	RESERVED	R	D	

Registre definujú špeciálne operácie navrhnutého grafického kontroléra. Každá operácia využíva 2 registre. Adresy pre registre sú rezervované v rozsahu x7FFF0-x7FFFB. Tabuľka tab. 3.3. obsahuje zoznam nastaviteľných hodnôt s nasledovným popisom:

Slovo RESERVED označuje extra bity, ktoré nemajú žiadnu funkciu. Je doporučené do týchto bitov zapísať nulovú hodnotu.

LINE – vykreslenie čiary, zadávané parametre sú súradnice počiatočného bodu X1,Y1, konečného bodu X2, Y2 a 24 bitová farba čiary reprezentovaná 8-bitovými farebnými hĺbkami RED, GREEN, BLUE. Nastavením bitu R (Request) do logickej 1 je operácia započatá a po ukončení operácie systém nastaví bit D (Done) na 1.

ROM – Zapísanie symbolu z ROM do zobrazovacej pamäti. Je nutné zadať súradnice počiatočného bodu X, Y, farbu RED, GREEN, BLUE a ID, alebo identifikačné číslo pamäte ktorá má byť použitá. Bity R a D slúžia na iniciovanie operácie a potvrdenie jej ukončenia.

BITBLT – Kopírovanie bloku zobrazovacej pamäti na zvolené miesto. Táto operácia vyžaduje súradnice troch bodov, počiatočný bod zdrojovej oblasti X1, Y1, posledný bod zdrojovej oblasti X2, Y2 a počiatočný bod cieľovej oblasti X3 a Y3. Ovládanie prebieha pomocou bitov R a D.

FILLRCT – vykreslenie obdĺžnika, parametre sú totožné s LINE registrom.

BRAMOP – operácie využívajúce BRAM pamäť čipu FPGA. Parametre v registri sú 17 bitové adresovanie ADDRESS, farebná mapa RED, GREEN, BLUE, bit masky M a mód operácie. Podporované módy zahŕňajú: zápis - 0b000, zápis masky - 0b001, zápis farby - 0b010, čítanie - 0b100, čítanie masky - 0b101, čítanie farby - 0b110. Bity R a D slúžia na kontrolu spúšťania funkcie.

BRAMCPY – kopírovanie medzi internou BRAM a zobrazovacou pamäťou, parametre sú analogické ku registru BITBLT, je nutné zadať súradnice prvého a posledného bodu kopírovanej oblasti z BRAM, BX1, BY1, BX2, BY2 a súradnice cieľového umiestnenia FBX, FBY. Bit M rozhoduje o použití masky, v prípade že je v logickej 0, maska nie je použitá. Iniciácia a ovládanie operácie je totožné s predchádzajúcimi registrami.

Funkcia BitBLT a využívanie BRAM nie je súčasťou zadania, jedná sa o nepovinné rozširujúce funkcie. BitBLT funkcia nie je implementovaná.

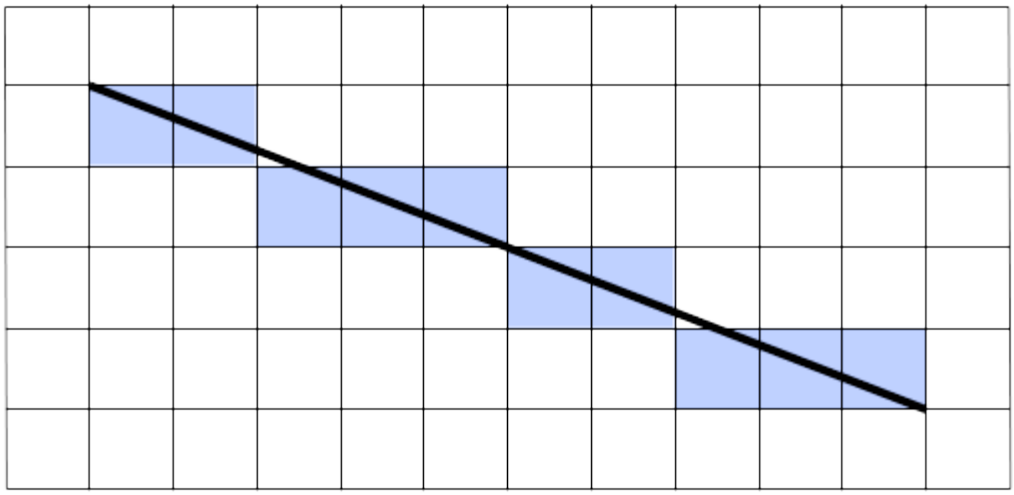
3.6 Vykresľovacia jednotka

Vykresľovacia jednotka má na starosti výpočty potrebné na generovanie 2D úsečiek. Dáta pre výpočty sú zasielané z registrovej časti obvodu. Na získanie jednotlivých bodov čiar využíva Bresenhamov algoritmus úsečky. Tento algoritmus vytvoril zamestnanec IBM, J. E. Bresenham v roku 1962. V súčasnosti existujú sofistikovanejšie algoritmy podporujúce antialiasing a rôzne transformácie, ale vzhľadom na obmedzené možnosti použitého FPGA ich aplikovanie neprináša žiadne výhody, naopak zvýšilo by sa zaťaženie dostupných prostriedkov. Bresenhamov algoritmus optimalizovaný pre procesory využíva iba celočíselné sčítanie, odčítanie, bitové posunutie a porovnávanie, čo ho robí vhodným aj pre použitie v hradlových poliach FPGA. Algoritmus počíta so zadaným počiatočným bodom úsečky [x_0, y_0] a konečným [x_1, y_1], pre ktoré platí rovnica úsečky: [15]

$$\frac{y-y_0}{y_1-y_0}=\frac{x-x_0}{x_1-x_0} \tag{1}$$

Výpočet následne prebieha tak, že sa pre postupne zvyšujúcu hodnotu x súradnice, hľadajú y súradnice ktoré majú odchýlku od skutočnej hodnoty medzi -0,5 a 0,5. So zvýšením hodnoty x sa zvýši chyba o podiel vzdialenosti jednotlivých súradníc koncových bodov vyjadrujúci strmosť čiary a v prípade, že prekročí hodnotu 0,5 je zmenšený o 1 zároveň so zvýšením hodnoty y o jedna, až kým nie je dosiahnutý finálny

bod [x_1, y_1]. Podľa smeru, ktorým je čiara vedená, sa počas inicializácie zamenia hodnoty x_0 a x_1 alebo sa namiesto inkrementácie použije dekrementácia hodnoty y. Odstránenie výpočtov obsahujúcich operácie s desatinnými číslami, náročných na výkon sa dosiahne vynásobením systému rozdielom x súradníc $|x_1 - x_0|$ a obrátením smeru počítania chyby tak, aby jej počiatočná hodnota začínala na úrovni 0,5 a postupne sa zmenšovala k nule. Výsledok je vektor súradníc reprezentujúci požadovanú úsečku. [15] Na obr. 3.5 je možné vidieť výsledok zobrazenej čiary oproti reálnej. Modifikovanú verzia tohto algoritmu je možné použiť na vykreslenie kružníc a elíps.



Obr. 3.5: Vykreslenie čiary pomocou Bresenhamovho algoritmu

Výpočet súradníc prebieha vo viacerých krokoch, nutné je pred-spracovanie počiatočných dát a vzhľadom na závislosť novej súradnice na predchádzajúcich sú možnosti paralelných výpočtov obmedzené. Aj napriek tomu je počet operácií v rámci jednej periódy hodinového signálu značne vyšší ako pri softvérovom riešení.

Vykresľovacia jednotka vypočítané súradnice konvertuje na adresu a postupne zasiela do vstupnej medzipamäte ako požiadavky na zápis. Počas práce vykresľovacej jednotky sú všetky ostatné funkcie pozastavené, až do dokončenia operácie. To umožňuje predchádzať konfliktom v prípade pokusu o prístup do zobrazovacej pamäte z viacerých zdrojov.

3.7 Jednotka pre komplexné pamäťové operácie

Úlohou jednotky pre komplexné pamäťové operácie je kopírovanie obdĺžnikových oblastí medzi jednotlivými časťami zobrazovacej pamäte, s prípadným využitím odlišných zdrojov pre vstupné dáta. V prvej fáze bude jednotka slúžiť na vykresľovanie dát uložených v pamäti typu ROM, konkrétne textových znakov. Dodatočne budú pridané funkcie na BitBLT operácie, kreslenie jednofarebných obdĺžnikov a využitie internej BRAM ako medzipamäte určenej na uloženie často opakovaných dát.

Zápis dát z ROM je nasledovný: v pamäti je uložená maska obrazca na základe ktorej sú

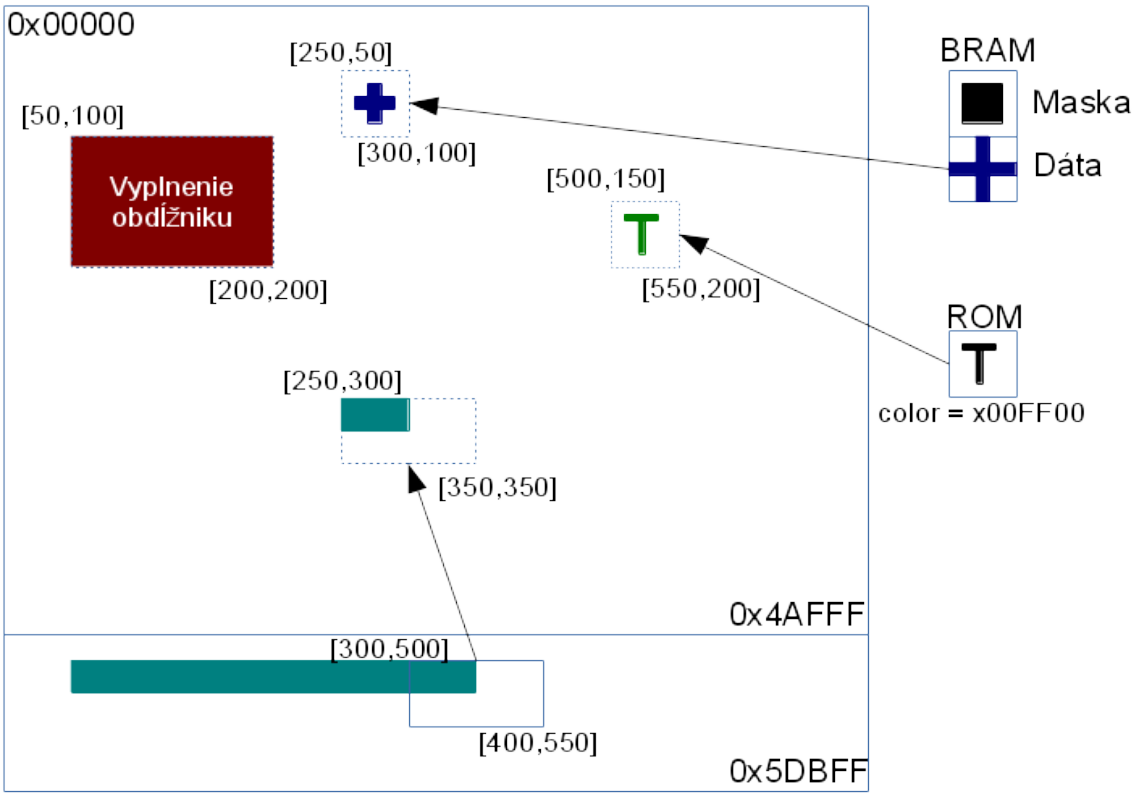
identifikované body, ktoré budú zapísané a body, ktoré budú ignorované. Farba je nastavená externe, spolu s počiatočným bodom a identifikačnou adresou vykresleného obrazca. Na konci každého riadku masky je adresa inkrementovaná o dĺžku riadku, v navrhovanej konfigurácií o 640. Dáta sú následne odoslané na zápis. Všetky ROM bunky obsahujú jeden symbol s pevne stanovenou veľkosťou.

Podobný princíp je použitý aj pri vypĺňaní obdĺžnikových oblastí. Zápis prebieha bez použitia masky, takže vstupné parametre sú počiatočný a posledný bod zápisu, definujúce rozmery a polohu vykresľovaného obdĺžnika a farbu výplne.

Pri BitBLT operáciách je časť pamäte kopírovaná na iné miesto. Vstupné parametre sú prvý, posledný pixel zdrojovej oblasti a prvý pixel cieľovej oblasti. Dáta sú postupne načítané zo zobrazovacej pamäte a znovu zapisované na novú adresu. V prípade potreby je možné uložiť dáta v slepej oblasti zobrazovacej pamäte a skopírovať ich na rôzne umiestnenia.

Využitie BRAM oproti štandardnej BitBLT operácii prináša hlavne lepší výkon a možnosť ukladania masiek obrazca. Nevýhodou je malá kapacita BRAM pamäte a tým limit veľkosti uložených obrazcov. Operácia funguje na podobnom princípe ako čítanie z ROM, avšak dáta nie sú prednastavené.

obr. 3.6 zobrazuje príklady operácií podporovaných jednotkou pre pamäťové operácie. Na obrázku je zobrazovacia pamäť v rozsahu 0x00000-0x5DBFF pre lepšiu názornosť. Maximálna reálna adresa je 0xFFFEE.



Obr. 3.6: Príklady jednotlivých pamäťových operácií.

3.8 Vstupná medzipamäť

Vstupná medzipamäť slúži na minimalizáciu čakacej doby pri zápise. Zásadný výkonnostný problém vykresľovacej pamäte je, že počas čítania dát na vykreslenie je prístup do nej limitovaný. V základe by to znamenalo obmedzenie možnosti zápisu na dobu zatemnenia a synchronizácie monitora. Ak by došlo ku požiadavke na zápis na začiatku vykresľovania riadku, systém by musel čakať na dokončenie zápisu 25,6 µs, čo je neprípustné, navyše by mohlo dôjsť pri rozsiahlejšej zmene obsahu vykresľovacej pamäte ku viditeľným chybám zobrazenia. Medzi možné riešenia tohto problému patria: zdvojenie zobrazovacej pamäte, využitie medzipamätí, vyššej rýchlosti jadra systému ako vykresľovania alebo použitie pamäte s oddeleným čítaním a zápisom.

Zdvojenie zobrazovacej pamäte (double buffering) znamená použitie dvoch a viac vykresľovacích pamätí rovnakej veľkostí. Zápis a čítanie tak prebiehajú nezávisle na sebe, kedy jedna pamäť je použitá pre zápis, druhá pre zobrazovanie. Pamäte sú následne podľa potreby synchronizované alebo multiplexované. Toto riešenie je z pohľadu latencie a korektného vykresľovania najoptimálnejšie, ale vyžaduje minimálne dvojnásobne veľkú vykresľovaciu pamäť s oddeleným ovládaním pamäťových čipov, takže je nepoužiteľné vzhľadom na použitý hardvér. Kompromis medzi rýchlosťou zápisu a hardvérovej

náročnosti je použitie medzipamätí. Súčasť tohto riešenia je práve vstupná medzipamäť. Ďalšie zlepšenie nastáva rozdielnym taktom jadra kontroléra a zobrazovania, kedy jadro pracuje na frekvencií 50 MHz a zobrazovací kontrolér na frekvencií 25 MHz. Zväčšenie tohto rozdielu je limitované použitým FPGA, rozsahom navrhnutého obvodu a použitým výstupným rozlíšením. Posledná spomenutá možnosť, pamäť s oddeleným zápisom a čítaním, je neaplikovateľná, vzhľadom na použité pamäťové čipy.

Jedná sa o pamäť typu FIFO (First In, First Out), s rozsahom 1024 záznamov o veľkosti 43 bitov. Každý záznam obsahuje 19-bitovú adresu zapisovaného pixelu a informácie o jeho farbe o veľkosti 24 bitov. Uložené dáta sú postupne odosielané na zápis do vykresľovacej pamäte. Ovládacie signály sú *full*, *empty* a *suspend*. Signál *full* oznamuje, že pamäť je plná a blokuje zápis do pamäte. Signál *empty* je aktívny, keď je pamäť prázdna a ukončuje sa ním odosielanie dát na ďalšie spracovanie. Signál *suspend* blokuje celú medzipamäť a je nastavený na logickú '1' v prípade že prebieha čítanie z nadradeného systému. To prebieha prioritne, aby bola doba čakania na dáta minimalizovaná.

3.9 Pamäťový kontrolér

Pamäťový kontrolér ovláda čítanie a zápis do hlavnej pamäte, zabraňuje konfliktom medzi rôznymi paralelnými požiadavkami a zaobstaráva čo najkratšie čakacie doby na pamäťové operácie. Jedná sa o generický radič, jeho komunikačné rozhranie nie je prispôsobené komunikácii s konkrétnym typom pamäte, preto je vo väčšine prípadov nevyhnutné použiť dodatočný ovládač, fungujúci ako komunikačná vrstva medzi pamäťou a radičom pamäte. Túto konfiguráciu využíva aj návrh testovacieho zapojenia.

Prioritou kontroléra je včasné odosielanie korektných dát na zobrazovanie. To je riešené pomocou interných čítačov, signalizácie nedostatku dát vo VGA medzipamäti a synchronizačných signálov z VGA kontroléra. Čítače reprezentujú poradie vo vykresľovanom riadku a samotnú adresu požadovaného pixelu. Po ohlásení nedostatku dát vo VGA medzipamäti je prioritne predaný prístup do pamäte čítaniu pre zobrazenie, okamžite po dokončení prebiehajúcej operácie. Synchronizačný signál *vga_line_sync* oznamuje koniec riadku a spúšťa po vynulovaní čítača pozície v riadku, načítavanie dát pre nový riadok. Signál *vga_sync* funguje podobne ako *vga_line_sync*, ale miesto konca riadku oznamuje ukončenie zobrazovania posledného pixelu na obrazovke. Po jeho prijatí dochádza ku resetu všetkých signálov a dát určených pre zobrazovanie. Externá synchronizácia bola zvolená preto, aby boli vylúčené potenciálne problémy s časovaním, a aj v prípade internej chyby je funkcia obnovená najneskôr po jednom zobrazovacom cykle. Ak nedochádza ku synchronizácii, ani VGA medzipamäť nehlási nedostatok dát, priorita čítania zobrazovacích dát je znížená a operácia prebieha, iba ak nie sú žiadne iné požiadavky na systém až do zaplnenia VGA medzipamäte alebo príchodu synchronizačných pulzov. Podporované je reťazené čítanie z pamäte, čo umožňuje čítanie v sekvenciách počas jednej periódy hodinového signálu.

Po prednostnom odosielaní dát na zobrazenie, v prípade stavu nedostatočného zaplnenia VGA medzipamäte, má najvyššiu prioritu požiadavka na prečítanie časti pamäte z nadradeného systému. Tá je predaná ovládaču konkrétnej pamäte. Po prečítaní dát je výsledok odoslaný späť do nadradeného systému. Odpovede sú generované asynchrónne, z dôvodu redukcie latencie systému. Podpora reťazenia požiadaviek zlepšuje rýchlosť odozvy pri viacnásobných čítacích sekvenciách. Čítacia sekvencia je prerušená buď procesom s väčšou prioritou alebo synchronizačnými signálmi, ktoré automaticky iniciujú po ukončení poslednej operácie počiatočné naplnenie VGA medzipamäte.

Posledná operácia, zápis do pamäte prebieha iba v prípade že neprebieha čítanie z pamäte, synchronizácia alebo prednostné čítanie pri zobrazovaní. Dáta sú uložené do registrov a odoslané ovládaču pre konkrétny typ pamäte. Ďalšie operácie neprebiehajú, kým nepríde potvrdenie ukončenia zápisu. Po jeho príchode sa proces opakuje. Tento prístup umožňuje rôznu dobu zápisu a čítania a zároveň podporu rôznych typov pamätí.

Možný zdroj konfliktov pri zápise a čítaní sú prechody medzi jednotlivými operáciami. Aby sa predišlo chybám pri týchto prechodoch, musia byť špecificky ošetrené, prechod môže nastať až po potvrdení z externého ovládača o dokončení poslednej operácie a všetky výstupné dáta sú ukladané do registrov. Realizácia modulu má formu komplexného stavového automatu.

3.10 VGA medzipamäť

Ďalší komponent slúžiaci na zníženie problému s odozvou spomenutom v kapitole 3.8 je VGA medzipamäť. Funkciou tejto pamäte je ukladať a predávať informácie o zobrazovaných pixeloch. Použitie tohto modulu značne znižuje latenciu pri jednotlivých operáciach iniciovaných nadradeným systémom, hlavne požiadaviek na čítanie, pri ktorých musí systém čakať na dokončenie poslednej operácie, kým môže odoslať ďalšiu. Medzipamäť medzi VGA kontrolérom a hlavnou zobrazovacou pamäťou poskytuje možnosť rozdelenia načítavania riadku na zobrazenie do viacerých sekvencií variabilnej dĺžky. Systém tak nemusí čakať na ukončenie zobrazovania aby splnil požiadavky vstupného systému. Je ale nevyhnutné zaistiť korektné zobrazovanie dát. V prípade nedostatku záznamov v pamäti je ohlásený nedostatočný stav radiču pamäte, ktorý začne prioritne načítavať dáta zo zobrazovacej pamäte. Aby sa zredukoval počet prechodov medzi operáciami, ktoré znižujú efektivitu systému vzhľadom na použitie reťazenia v ovládači pamäte, prechod do stavu nedostatočného počtu záznamov a ukončenie tohto stavu nastavajú pri rôznych úrovniach zaplnenia modulu.

VGA medzipamäť je vytvorená pomocou internej BRAM pamäte integrovanej v FPGA. Jedná sa o pamäť typu FIFO, takže prvý uložený záznam je zároveň aj prvý odoslaný záznam.

Sekundárna funkcia VGA medzipamäte je oddelenie modulu VGA kontroléra

používajúceho hodinový signál 25 MHz od zvyšku systému, ktorý môže pracovať na odlišnej frekvencií. V testovacej implementácií je použitý 50 MHz hodinový signál.

Veľkosť pamäte je maximálne 640 záznamov, čo zabraňuje potenciálnemu pretečeniu dát medzi riadkami a chybám pri konflikte ukazovateľov zápis a čítania. Reálna komponenta je realizovaná pomocou 10-bitovej pamäte s kapacitou 1024 záznamov, prebytočné voľné miesto je tak nevyužité, ale zväčšenie kapacity VGA medzipamäte v návrhu by neprinieslo žiadne zlepšenie výkonu systému, naopak vyžadovalo by komplexnejšiu ovládaciu logiku.

Pre zaistenie korektného časovania a funkcie VGA medzipamäte aj v prípade nepredvídateľných problémov, sú ukazovatele zápisu a čítania asynchrónne vynulované vždy po príchode synchronizačných signálov z VGA kontroléra. Tieto signály majú na modul rovnaký efekt ako hlavný reset. Synchronizačné signály *vga_line_sync* a *vga_sync* sú odosielané vždy po dokončení zobrazovania riadku, respektíve zobrazenia posledného pixelu na obrazovke.

3.11 VGA kontrolér

Správny výstup z VGA portu má za úlohu generovať VGA kontrolér. Ten sa skladá z časovacích signálov a dátových signálov reprezentujúcich jednotlivé farby. Navrhnuté grafické jadro používa 24-bitovú farebnú hĺbku a časovanie pre rozlíšenie 640×480 pri obnovovacej frekvencii 60 HZ definované vo VESA štandarde. [16] Časovanie sa nachádza v tab.3.4. Kontrolér pracuje na frekvencii 25 MHz, generovanej externým deličom signálu zo vstupného 50 MHz hodinového signálu.

Tab. 3.4: Časovanie pre zvolený zobrazovací mód. [16]

	Horizontálna synchronizácia		Vertikálna synchronizácia	
	Počet bodov	Čas (µs)	Počet riadkov	Čas (ms)
Oblasť zobrazenia	640	25,422	480	15,253
Predná doba zatemnenia	16	0,635	10	0,318
Synchronizačný pulz	96	3,813	2	0,064
Spätná doba zatemnenia	48	1,907	33	1,049
Celý cyklus	800	31,778	525	16,683

Časovacie signály sú *h_sync* pre horizontálnu synchronizáciu a *v_sync* pre vertikálnu synchronizáciu. Tieto signály sú tvorené pomocou dvoch navzájom nezávislých čítačov. Časovanie sa skladá z doby zobrazovania, kedy je zobrazovaný obraz na monitore, prednej doby zatemnenia, spätnej doby zatemnenia, kedy je odosielaný synchronizačný signál ale dáta nie sú zobrazované a synchronizačného pulzu, kedy sa je synchronizačný signál nastavený na logickú nulu a odosielané vektory reprezentujúce farby majú tiež nulovú hodnotu.

Farebné vektory s rozsahom 8-bitov na farbu zo škály RGB sú synchrónne načítavané z VGA medzipamäte. RGB farebná škála sa skladá z 3 farieb – červenej, zelenej a modrej, pričom výsledok je kombinácia intenzít týchto farieb. Každý pixel je v pamäti samostatne uložený a má vlastnú adresu, ktorá sa pri načítavaní postupne inkrementuje. VGA kontrolér s adresami nepracuje a automaticky predpokladá, že prijaté dáta správne reprezentujú zobrazovaný pixel.

Na korekciu prípadných chýb časovania a nesúladu odoslanej adresy pixelu so zobrazovanou pozíciou sú generované synchronizačné signály pre ostatné moduly systému, konkrétne VGA medzipamäť a radič pamäte.

3.12 Generátor vstupných dát

Generátor vstupných dát emuluje nadradený procesor. Jeho úloha je odosielať požiadavky cez zbernicu CLB na základe dát uložených v pamäti ROM.

Testovacie zapojenie so skutočným procesorom je neimplementovateľné pre zvolený vývojový kit, vzhľadom na jeho nedostatočné zdroje. Požadovaná plocha komplexného 32-bitového procesora Codix RISC, pre ktorý je grafický kontrolér navrhnutý, zaberá veľkú časť čipu Spartan 3, po jeho pripojení by celý systém vyžadoval viac ako 100% LUT komponentov v čipe. Použitie menších soft procesorov, napríklad 8-bitového PicoBlaze od firmy Xilinx je teoreticky možné avšak vyžaduje komplexnú prídavnú logiku pre funkčnú implementáciu komunikačného rozhrania cez CLB zbernicu a komplikovaný program riadiaci chovanie procesoru. Použitie jednoúčelového modulu zameraného na čítanie z ROM pamäti a generovanie kontrolných signálov umožňuje najflexibilnejšie testovanie a je nenáročné na prostriedky čipu.

Testovacia sekvencia je nasledovná: v prvej fáze sa naplní celá zobrazovacia pamäť testovacími dátami. Tieto dáta vytvárajú obrazec, skladajúci sa zo štyroch odlišných oblastí obdĺžnikového tvaru s rozlíšením 320×240. Následne sú tieto dáta spätne prečítané. Táto fáza slúži primárne na verifikáciu počas simulácie, pri reálnom zapojení v použitej vývojovej doske nie je možné výstup kontrolovať. Tretia časť testovacej sekvencie spočíva v prepísaní časti zobrazovacej pamäte, prvých dvoch riadkov dátovými vektormi zloženými z logických jednotiek. Zmena by sa mala prejaviť prefarbením daných pixelov na monitore na bielo. Na zakončenie testovania práce s pamäťou sú prečítané dáta z náhodnej pamäťovej pozície a časť získaného vektoru zobrazená pomocou 8 LED diód na vývojovej doske. Zvyšok testov testuje zápis do registrov a funkčnosť akceleračných funkcií.

3.13 SRAM kontrolér

Aby bola dosiahnutá variabilnosť návrhu samotného grafického kontroléra, radič pamäte má značne generické komunikačné rozhranie pre ovládanie pamäte. SRAM

kontrolér je vrstva zaisťujúca kompatibilitu tohto rozhrania a SRAM pamäte použitej v testovacom zapojení.

Radič pracuje s dvoma SRAM čipmi s kapacitou 512 Kb, latenciou do 10 ns a asynchrónnym prístupom. Tieto čipy majú zdieľanú adresu, signál pre ovládanie zápisu *we* a signál pre ovládanie čítania *oe*. Samostatne pripojené ku obvodu FPGA sú *ce* signály ktoré slúžia na aktiváciu čipu, trojvstavová vstupno-výstupná 16 bitová zbernica a signály *ub*,*lb* pre výber konkrétneho bajtu z výstupu. Všetky kontrolné signály sú invertované, čo znamená že sú aktívne na úrovní v logickej 0 a neaktívne pri logickej 1.

Adresovanie má rozsah 18 bitov. Správna funkcia grafického jadra vyžaduje aspoň 19 bitový adresný priestor, preto je 19. bit emulovaný pomocou *ce* signálov. Prepínanie medzi jednotlivými čipmi vyžaduje špeciálnu pozornosť, hlavne pri zreťazenom čítaní, kedy napríklad pri zmene adresy z 0x3FFFF na 0x40000 musia byť aktívne obidva čipy, čip poskytujúci dáta aj čip, v ktorom je nastavovaná nová adresa. Pri zápise tento problém nenastáva, nedochádza ku reťazeniu požiadaviek a nové informácie sa na výstup dostávajú až po ukončení prechádzajúcej operácie.

Radič rovnako zabraňuje konfliktom pri zmene typu operácie z čítania na zápis alebo naopak. To je zariadené pozastavením všetkých operácií na 1 periódu hodinového signálu, kým je začatá nová operácia.

Čítanie prebieha prioritne v súlade s funkciou zvyšku systému, avšak ku paralelným požiadavkám by dochádzať nemalo. Sekvencie viacerých príkazov na čítanie majú podporu reťazenia, takže prvá operácia trvá 2 cykly hodinového signálu, ostatné jeden. Čítanie prebieha asynchrónne, výstup je odoslaný okamžite po získaní dát.

Zápis prebieha v dvoch fázach. V prvej sa nastaví adresa, dáta a signál *we* na výstupy, v druhej je dokončený *we* impulz a je späť nastavený na úroveň logickej 1. Na dosiahnutie stability zápisu sú výstupné signály oddelené registrami. Zápis tak prebieha v dvoch periódach hodinového signálu. Napriek použitu asynchrónnej pamäte je zápis synchrónny, aby sa predišlo problému s časovaním a nekorektným zápisom z neho vyplývajúcim.

3.14 VGA konvertor

VGA konvertor je jeden z komponentov, ktoré nie sú súčasťou samotného grafického kontroléra, ale slúžia na dosiahnutie kompatibility návrhu s použitou hardvérovou konfiguráciou. 24 bitový výstup navrhnutého grafického jadra je nutné previesť na 3 bitový, vzhľadom na absenciu RAMDAC čipov na testovacej doske plošných spojov. Samotný prevod prebieha s minimálnymi nárokmi na zapojenie – najvýznamnejší bit z každej farby zo škály RGB, reprezentovanej 8 bitovým vektorom je predaný ďalej a ostatné bity sú zahodené. Tento typ konverzie poskytuje relatívne korektné výsledky vzhľadom na rozsah redukcie farebného spektra a zároveň nevyžaduje žiadnu dodatočnú logiku.

3.15 Možnosti rozšírenia grafického jadra

Návrh počíta s potenciálnym rozširovaním grafického jadra. Modulárny prístup umožňuje výmenu jednotlivých komponentov alebo relatívne jednoduché pridávanie nových. Hlavným problémom pri rozširovaní je použitie špecifických hodinových signálov a optimalizácia pre konkrétny obvod FPGA. V prípade použitia Spartan 3 XC3S200 čipu je limitujúcim faktorm veľkosť návrhu.

Podpora viacerých zobrazovacích módov by vyžadovala relatívne malé zmeny vo VGA kontroléri a niektorých ďalších komponentoch ale bola by možná spolu s dynamickým menením rozlíšenia. Použitie lineárneho adresovania zobrazovacej pamäte značne uľahčuje prechod medzi zobrazovacími módmi. Veľkosť, rýchlosť externej pamäte a frekvencia hodinového signálu sú parametre od ktorých najviac závisí maximálne možné použité rozlíšenie a obnovovacia frekvencia. Je pravdepodobné, že pre väčší set zobrazovacích módov, by bolo nevyhnutné zaviesť nejaký spôsob dynamického taktovania obvodu.

Pridávanie nových akceleračných funkcií je možné jednoduchým rozšírením registrov a pridaním nových funkčných modulov. Rovnako by bolo možné výpočty paralelizovať zdvojením logiky, ale pokiaľ by systém nepracoval s extrémne rýchlou pamäťou nemá tento prístup význam. Problematické by bolo aj zavedenie viacnásobného bufferingu, vyžadovalo by to úplnú zmenu spôsobu práce s pamäťou.

Zmena komunikačných rozhraní je možná zavedením nových periférnych ovládačov, ale tie musia byť nevyhnutne upravené pre špecifické potreby grafického jadra alebo využívať špeciálnu medzivrstvu. Štandardné radiče je možné pripojiť ku navrhnutému kontroléru.

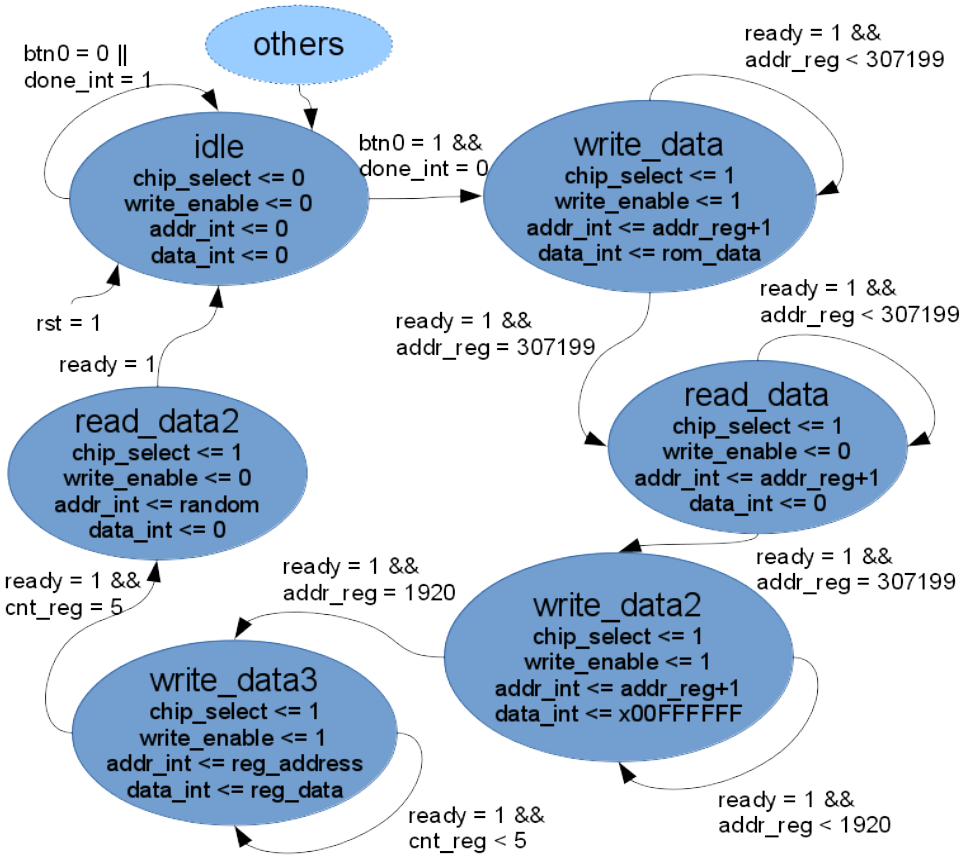
4 Implementácia

4.1 Modul top

Top je štrukturálny modul najvyššej úrovne v ktorom sú prepojené jednotlivé komponenty. Neobsahuje žiadnu vlastnú logiku ani registre. Interné komponenty sú generátor vstupných dát *input_generator*, samotné grafické jadro *fbd_core*, radič pamäťových čipov *sram_driver* a VGA konvertor *vga_converter*. Modul top je buď pripojený do testovacieho modulu alebo po pridaní UCF (User Constraint File) súboru s informáciami o mapovaní pinov na FPGA spracovaný nástrojmi na syntézu, preklad, mapovanie, rozmiestňovanie a prepojenie komponentov a nahraný priamo do FPGA cez rozhranie JTAG. Zapojenie jednotlivých modulov zodpovedá schéme na obr. 3.2 v kapitole 3.3. Názvy použité pri popisovaní stavov, signálov a modulov v obrázkoch sú preberané z VHDL kódu.

4.2 Modul input_generator

Modul *input_generator* je funkčná implementácia navrhnutého generátoru vstupných dát z kapitoly Návrh systému. Jedná sa o sekvenčný obvod obsahujúci kombinačnú logiku a asynchrónny reset. Systém je tvorený stavovým automatom, v ktorom každý stav reprezentuje jednu testovaciu sekvenciu a jednoduchou logikou na emuláciu CLB master bloku. Táto logika spočíva v predávaní registrov na výstupy modulu a generovania *ready* signálu na základe AACK odpovede z podriadeného bloku. Stavový automat obsahuje stavy *idle*, *write_data*, *read_data*, *write_data2*, *read_data2* a *write_data3*. Prechody a nastavované signály sú na obr. 4.1. Priebeh testovacej sekvencie je možné spustiť maximálne jedenkrát od posledného resetu, aby bolo možné odhaliť neúplný zápis počas jedného zapisovacieho cyklu.



Obr. 4.1: Stavový automat reprezentujúci testovaciu sekvenciu.

Stav *idle* je základný stav, do ktorého automat prechádza po resete, v prípade chyby alebo dokončení testovacej sekvencie. Všetky hodnoty sú vynulované, s výnimkou signálu *done_int*, ktorého nastavenie na logickú 1 znamená ukončenie testovacej sekvencie. Hodnoty sa nemenia až kým sa neobjaví externý podnet, v tomto prípade stlačenie tlačítka btn0, iniciujúceho prechod do stavu *write_data*, alebo reset resetujúci signál *done_int*.

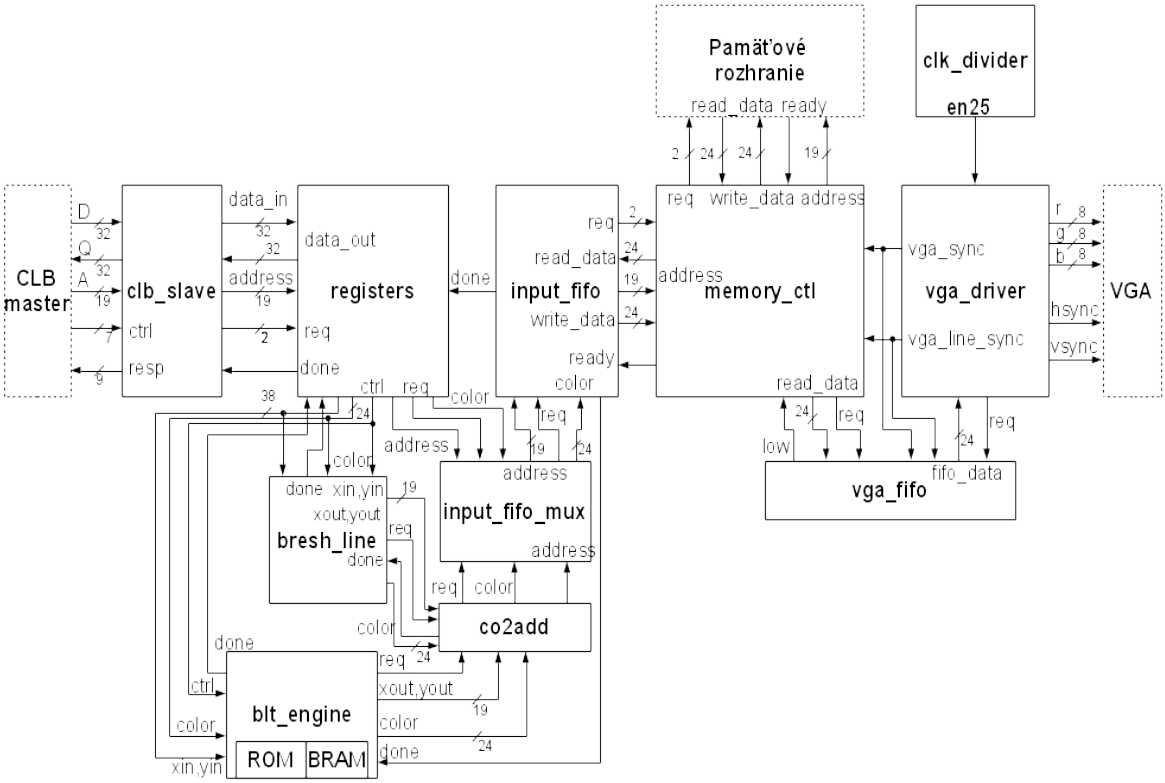
Počas stavu *write_data* sú postupne zasielané dáta na zápis na základe štyroch rom pamätí o veľkosti šestnásť 32 bitových záznamov. Odosielanie prebieha cyklicky, pričom konkrétna ROM pamäť je zvolená na základe registra adresy *address_reg*, tak aby vznikol v každom kvadrante obrazovky samostatný periodicky sa opakujúci obrazec. Adresný register je postupne inkrementovaný vždy po úspešnom zapísaní predchádzajúceho pixelu. V momente keď je zapísaný posledný bod na obrazovke s adresou 0x4AFFF je *write_data* ukončený.

Druhá testovacia sekvencia, postupné načítanie celého obsahu pamäte prebieha v stave *read_data*. Adresný vektor je opäť postupne inkrementovaný na základe odozvy grafického jadra. Prijaté dáta nie sú ukladané, ich korektnosť nie je možné v reálnej aplikácií overiť. Táto sekvencia slúži primárne pre účely simulácie. Po prečítaní dát z adresy 0x4AFFF prechádza stavový automat do stavu *write_data2*.

Ten je podobný ako stav *write_data*. Opäť sa zapisujú dáta do pamäte grafického jadra,

tentokrát ale len 2 riadky a všetky body bielej farby. Po zapísaní posledného pixelu na pozícií 1379 sa spúšťa posledný test pamäte, reprezentovaný stavom *read_data2* v ktorom sú prečítané dáta z náhodne zvolenej adresy a časť z nich zobrazená na 8 LED diódach.

4.3 Grafické jadro fbd_core

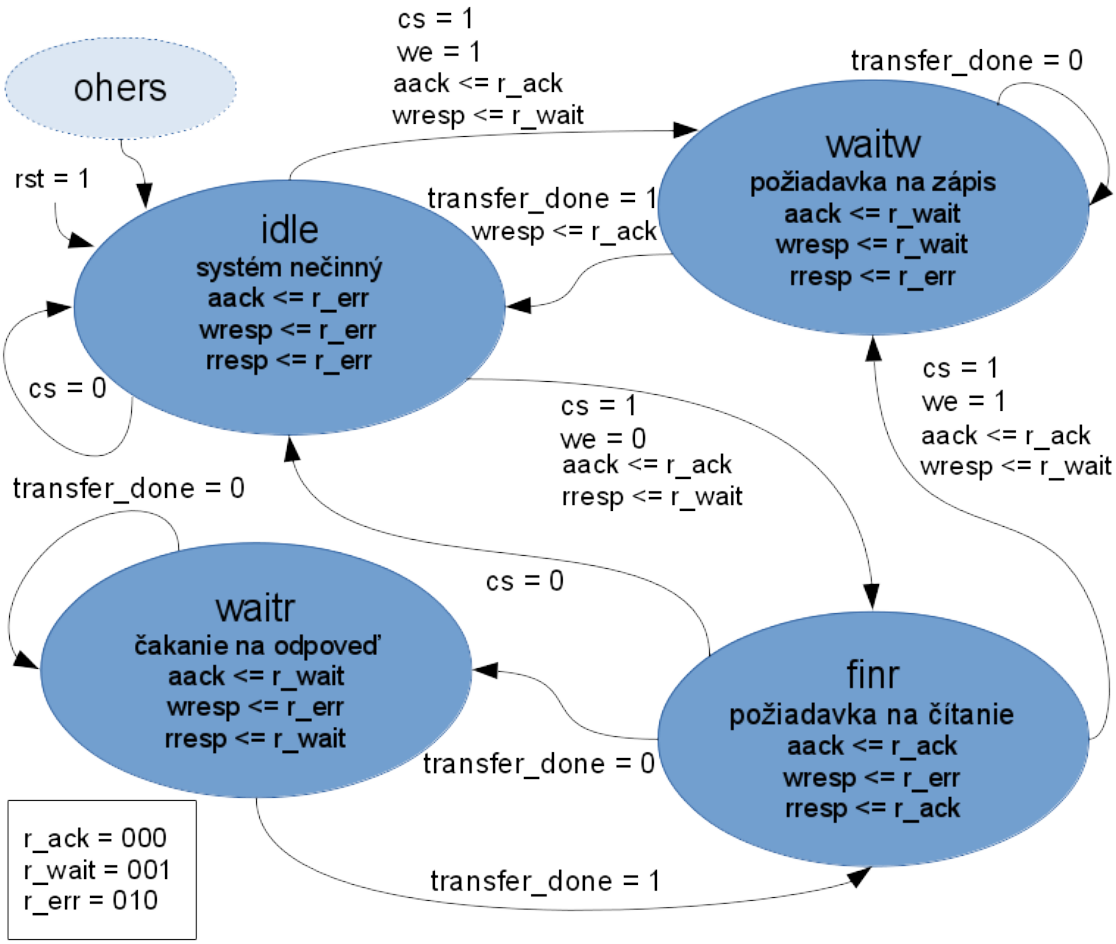


Obr. 4.2: Bloková schéma grafického jadra fbd_core

Tento modul reprezentuje samotné jadro grafického kontroléra. Podobne ako modul top, ktorý je najvyšším modulom celého systému, *fbd_core* obsahuje iba deklaráciu komponentov, ich zapojenie a komunikačné signály. Jedná sa o štrukturálny popis interného zapojenia, jediná dodatočná logika je použitie operácie and na výstupy z akceleračných jednotiek. V jednom momente môže byť aktívny maximálne jeden výstup preto je daná konfigurácia dostatočná. Zapojenie komponentov v reálnom obvode po syntéze a nahraní do FPGA sa môže líšiť. optimalizačné nástroje často prispôsobujú štruktúru obvodu pre maximalizáciu efektivity využitia možností konkrétneho čipu. Grafické jadro *fbd_core* sa skladá z nasledujúcich komponentov: komunikačné rozhranie pre zbernicu CLB *clb_slave*, registre systému *registers*, vstupná medzipamäť *input_fifo*, radič pamäte *memory_ctl*, VGA medzipamäť *vga_fifo,* VGA kontrolér *vga_driver*, multiplexor *input_fifo_mux*, konvertor súradníc na adresu *co2add*, delička hodinového signálu *clk_divider* a akceleračné jednotky *mem_ops* a *bresh_line* Ich zapojenie je znázornené na obr. 4.2. Názvy signálov na obrázku sú odvodené zo signálov použitých vo VHDL kóde pomocou skracovania a zlučovania pre zlepšenie prehľadnosti obrázku.

4.4 Komunikačné rozhranie clb_slave

Clb_slave je modul slúžiaci na komunikáciu cez zbernicu CLB. Jeho úlohou je zasielať odpovede na príkazy z nadradeného systému a prenos dát medzi týmto systémom a zvyškom grafického jadra. Podporované sú zápis a čítanie. Implementácia má formu sekvenčného obvodu s asynchrónnym resetom a relatívne rozsiahlou kombinačnou časťou. Do registrov sú ukladané iba dátové vektory *d*, *q*, adresa *addr* a kódovanie stavového automatu. Výstupy *aack*, *wresp* a *rresp* sú riešené kombinačne, aby bolo umožnené reťazenie požiadaviek a nedochádzalo ku oneskoreným odpovediam.



Obr. 4.3: Stavový automat popisujúci chovanie modulu clb_slave

Stavový automat riadiaci odpovede nadradenému procesoru je zobrazený na obr. 4.3. Ak nie je nastavený signál *cs*, systém sa nachádza v pokojovom stave *idle*. Všetky výstupy odpovedí sú nastavené na úroveň 010 reprezentujúcu chybu. Do *idle* prechádza systém aj v prípade chyby interných signálov alebo resetu. V momente nastavenia *cs* signálu je dekódovaný vstup *we*, podľa ktorého sa určuje typ požiadavky – pri logickej 1 sa jedná o zápis a systém prechádza do stavu *waitw*, pri opačnej hodnote je začaté čítanie stavom *finr*. Signál *aack* potvrdzuje prijatie dát a odoslanie požiadavky nastavením sa na úroveň 000.

Počas *waitw* sú prijaté dáta odoslané na zápis. Signály *aack* a *wresp* majú hodnotu 001, reprezentujúcu hlásenie neukončenej požiadavky. Po potvrdení dokončenia zápisu sa systém vracia do stavu *idle* a signál *wresp* potvrdzuje koniec zápisu nastavením na úroveň 000.

Čítanie vyžaduje použitie dvoch stavov *waitr* a *finr*. *Waitr* je analogický ku *waitw,* ale namiesto zápisu sa čaká na ukončenie čítania a nastavuje sa signál *rresp*. Stav *finr* slúži na podporu reťazenia požiadaviek a predanie dát po ukončení čítania. Z tohto stavu je možné okamžite prejsť na zápis.

Komunikácia so systémovými registrami, ktoré poslané dáta ďalej spracúvajú, prebieha ukončenie transferu *transfer_done*. Dáta a adresa sú obojsmerne posielané bez dodatočných zmien.

4.5 Systémové registre

Hlavnú časť modulu tvoria registre grafického kontroléra. Tieto registre majú formu pamäte, tvorenej pomocou LUT tabuliek. Registre slúžia na spúšťanie akceleračných funkcií a nastavovanie ich parametrov.

Súčasť modulu je logika na dekódovanie pamäťovej adresy, na základe ktorej sa určí, či je požiadavka smerovaná na registre, alebo sa jedná o priamy prístup do zobrazovacej pamäte. Zápis a čítanie z registrov prebieha okamžite. V prípade že adresa smeruje na zobrazovaciu pamäť, najvýznamnejší bajt z dátového registra je ignorovaný, keďže systém pracuje s 24 bitovým farebným rozsahom. Operácie s priamym prístupom do pamäte sú odosielané do vstupnej medzipamäte, s ktorou komunikácia prebieha pomocou 4 kontrolných signálov: požiadavka na zápis, požiadavka na čítanie, ukončenie čítania a úplné zaplnenie pamäte. Obe operácie, zápis aj čítanie, sú riešené v samostatných procesoch, pričom čítanie má vyššiu prioritu ako zápis, aj keď stav, kedy prídu obidve požiadavky v rovnakom čase, by za normálnych okolností nemal nastať.

Operácie s registrami prebiehajú v priebehu jednej periódy hodinového signálu. Nadradený systém zapisuje na jednotlivé pozície parametre požadovaných funkcií. V prípade nastavenia jedného z R bitov z tabuľky tab. 3.3 je spustená príslušná akceleračná operácia. Implementácia podporuje funkcie vykreslenie čiary LINE, vyplnenie obdĺžnikovej oblasti FILLRCT, kopírovanie dát z ROM pamäti ROM, kopírovanie z BRAM pamäti BRAMCPY a zápis do BRAM pamäti BRAMOP.

LINE a FILLRCT zdieľajú vstupné parametre, registre odosielajú vektory súradníc *x1, x2, y1, y2*, farebnú hĺbku *color_data* a ovládacie signály *ctrl* a *line_req*. Funkcia ROM nevyžaduje zadanie koncového bodu vykresleného objektu, preto je zbernica *y2* nepoužitá a do *x2* je uložené identifikačné číslo požadovaného znaku. Identifikačné číslo je zhodné s ASCII (American Standard Code for Information Interchange) hodnotou znaku. Parametre BRAMOP sú súradnice miesta zápisu do BRAM, farbra zapísaného pixelu

a mód zápisu. Akceptované hodnoty pre mód zápisu sú 0b01 – zápis masky a dát, 0b010 - zápis výhradne dát a 0b011 – zápis masky. Na prenos dát sú použité výstupy *x1*, *y1* a *color_data* Pre vykonanie BRAMCPY je nutné zadať počiatočný a konečný bod kopírovanej oblasti z BRAM, nastaviť bit použitia masky a počiatočný bod cieľovej oblasti. Cieľové súradnice sa odosielajú pomocou signálov *x1* a *y1*, ostatné dáta sú uložnéné vo vektore *color_data*. V priebehu vykonávania akceleračnej operácie grafický kontrolér neprijíma vstupné príkazy až do jej ukončenia, aby sa predišlo konfliktom pri zápise z viacerých zdrojov. Po prijatí potvrdenia ukončenia operácie sú bity R a D prepísané na hodnoty 0 a 1.

4.6 Vykresľovací modul

Jedná sa o funkčný blok, ktorý ktorý z informácie o krajných bodoch čiary generuje dáta, reprezentujúce jednotlivé pixely v zobrazovacej pamäti. Modul využíva Bresenhamov algoritmus aproximácie úsečiek s miernymi modifikáciami pre využitie v obvodoch FPGA. *Bresh_line* je navrhnutý ako stavový automat, ktorý čaká na príkaz na vykreslenie čiary a na potrebné dáta. Tie sa skladajú zo súradníc koncových bodov *x1*, *y1*, *x2* , *y2* a vektoru farebnej hĺbky *color*. Výpočet prebieha v štyroch fázach, reprezentovaných stavmi automatu.

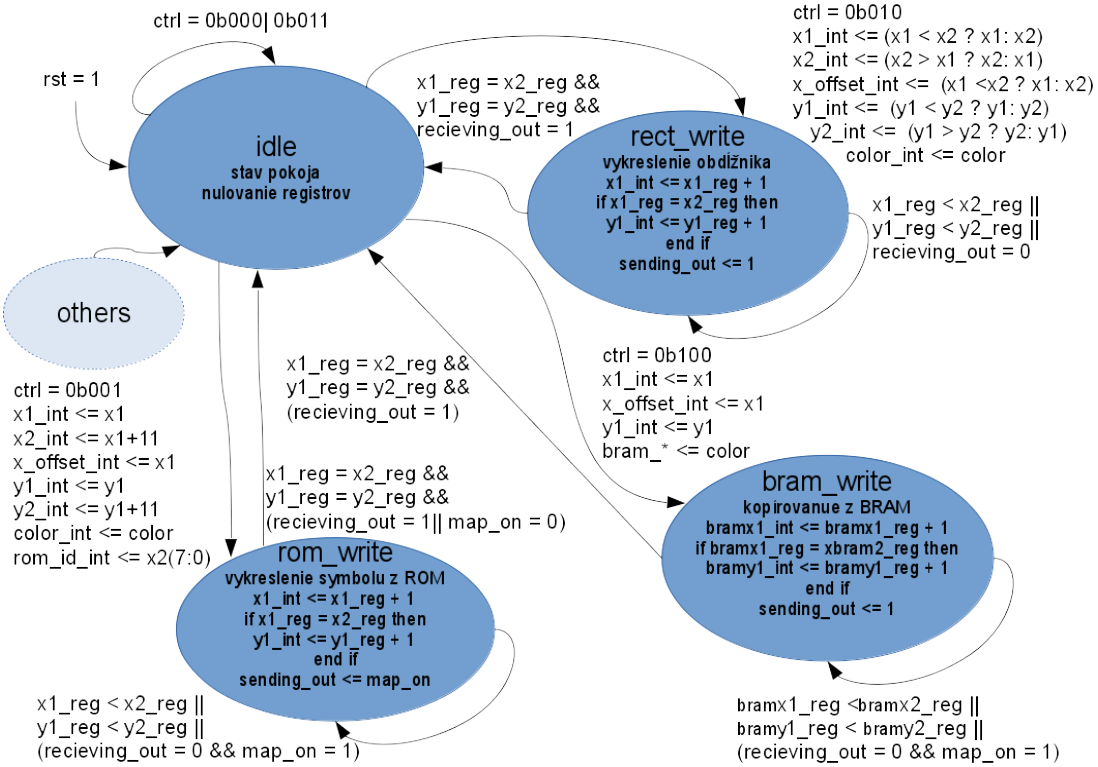
Po prijatí príkazu na operáciu prebehne príprava na samotný výpočet v stave *prep*. Pomocou komparátorov je určený smer úsečky a odčítaním sú získané rozdielové hodnoty počiatočných a konečných súradníc *dx* a *dy*. Počas nasledujúcej periódy hodinového signálu automat prechádza do stavu *errc*, v ktorom je vypočítané počiatočná hodnota chyby *err* podľa vzorca:

$$err = |xx1 - xx2| + |yy1 - yy2| \qquad (2)$$

Okrem toho je odoslaný na zápis prvý bod čiary so súradnicami [x1, y1]. Tým je ukončená príprava a začína cyklus počítania súradníc v stavoch *calcx* a *calcy*. Ak je dvojnásobok veľkosti chyby väčší ako rozdiel *dy* je súradnica x reprezentovaná signálom *xx1* zmenená o jedna, podľa smeru úsečky a veľkosť chyby znova prepočítaná. Analogický výpočet prebieha pri počítaní súradnice y, signálu *yy1*. Počas stavu *calcy* sú vypočítané súradnice odoslané do konvertora *co2add*. Tento proces prebieha až kým nie je odoslaná posledná súradnica [x2, y2]. Systém sa následne prepne do stavu *idle*. Vektor farby je uložený v registri *color_reg* a odosielaný s každou adresou. Viacfarebné čiary nie sú podporované. V čase vykresľovania systém neprijíma ďalšie príkazy, výstup *data_ready* je nastavený na 0. V prípade, že je nasledujúci člen zaneprázdnený alebo z nejakého iného dôvodu neprijíma dáta, systém ostáva v poslednom stave kým nie je možné odoslať ďalšie informácie o vykresľovanom bode. Jednotka je vybavená asynchrónnym resetom.

4.7 Modul pre pamäťové operácie mem_ops

Modul slúži na vykonávanie akceleračných operácií, požadujúcich prácu s väčšími blokmi pamäte. Podporované sú: vyplnenie obdĺžnikovej oblasti zvolenou farbou a kopírovanie dát z ROM pamäte. Modul je realizovaný pomocou stavového automatu, zobrazeného na obr. 4.4, s asynchrónnym resetom so *stavmi idle*, *rom_write, rect_write* a *bram_write*.



Obr. 4.4: Stavový automat modulu mem_ops

Základný stav je *idle*, systém je nečinný a vyčkáva na vstupné dáta. V prípade nastavenia signálu *ctrl* je zvolený nasledujúci stav na základe jeho hodnoty. Ak je *ctrl* nastavený na 0b010, nasledujúci stav je *rect_write*, registre *x1_reg*, *x2_reg*, *x_offset*, *y1_reg*, *y2_reg*, *color_reg* sú naplnené vstupnými dátami a je určený smer zápisu na základe zadaných súradníc. Stavový automat prechádza do stavu *rom_write*, ak je *ctrl* rovný hodnote 0b001. Načítané registre sú totožné ale je pridané identifikačné číslo znaku *rom_id_reg* a súradnice *x2_reg* a *y2_reg* sú odvodené zo vstupov *x1* a *y1* podľa vzorca:

$$\begin{aligned} x2\_reg &= x1 + 11 \\ y2\_reg &= y1 + 11 \end{aligned} \tag{3}$$

Nastavením signálu *ctrl* na hodnotu 0b100 prechádza automat do stavu *bram_write*. Dochádza ku zaplneniu registrov bram súradníc, *map_reg*, *x1* a *y2*. Počas stavu *idle* prebieha zároveň obsluha zápisu do BRAM pamäte ak *ctrl* má hodnotu 0b011. Vľkosť

BRAM je 32×32 údajov s rozsahom 25 bitov. Na základe nastavení registrov je zapísaný mapový bit, dátový vektor reprezentujúci farbu alebo oba údaje. Doba zápisu je nižšia ako jedna perióda hodinového signálu, preto by nebolo efektívne použiť samostatný stav.

V stave *rect_write* prebieha vykreslenie jednofarebného obdĺžnika. Odosielanie dát na zápis prebieha postupne, na základe súradníc uložených v registroch. Po prijatí potvrdenia zápisu sa *x1_reg* inkrementuje o jedna, až kým nemá rovnakú hodnotu ako *x2_reg*. Vtedy je *x1_reg* nastavený na úroveň *x_offset_reg* a ku *y1_reg* je pripočítaná 1. Ak sa hodnota *x1_reg* rovná *x2_reg* a *y1_reg* rovná *y2_reg*, systém odošle potvrdenie o ukončení operácie a vracia sa do stavu *idle*.

Načítanie z ROM pamäte prebieha v stave *rom_write*. Algoritmus zápisu je analogický ku vykresleniu obdĺžnika, ale symboly majú konštantnú veľkosť 12×12 a pixely sú zapísané iba v prípade, že je príslušný bit v mape rovný logickej 1. ROM dáta sú extrahované z bitových máp fontu DejaVu Sans o veľkosti 12 bodov a uložené do BRAM pamäte. Podporované sú základné znaky abecedy a číslice. Adresovanie je 12 bitové, horných 8 bitov reprezentuje ASCII hodnotu znaku uloženú v *rom_id_reg*, zvyšné 4 bity vyberajú 12 bitový riadok mapy. Po dokončení odosielania dát sa systém vracia do stavu *idle*.

Stav *bram_write* slúži na zápis dát z pamäti BRAM do zobrazovacej pamäti. Použitý algoritmus je odvodený zo stavov *rom_write* a *rect_write*. Zvolený obdĺžnik je odosielaný postupnou inkrementáciou súradníc ako v stave *rect_write* na zápis, na základe zadaných súradníc a dát uložených v BRAM. Podporovaná je variabilná veľkosť kopírovanej oblasti a voliteľné použitie zobrazovacej mapy.

4.8 Konvertor súradníc na adresu co2add

Vykresľovanie objektov a kopírovanie obdĺžnikových častí pamäte má značne nižšie nároky na veľkosť logiky, ak sa pracuje so súradnicami miesto konkrétnych adries. Preto sú dáta pre akceleračné funkcie zadávané v súradnicovej forme. Následne je nutné výsledok skonvertovať, aby boli vypočítané pixely požadovaných objektov správne zapísané do zobrazovacej pamäte. Konverzia zahŕňa vynásobenie súradnice y veľkosťou riadku x_{max} a pripočítanie súradnice x, podľa vzorca:

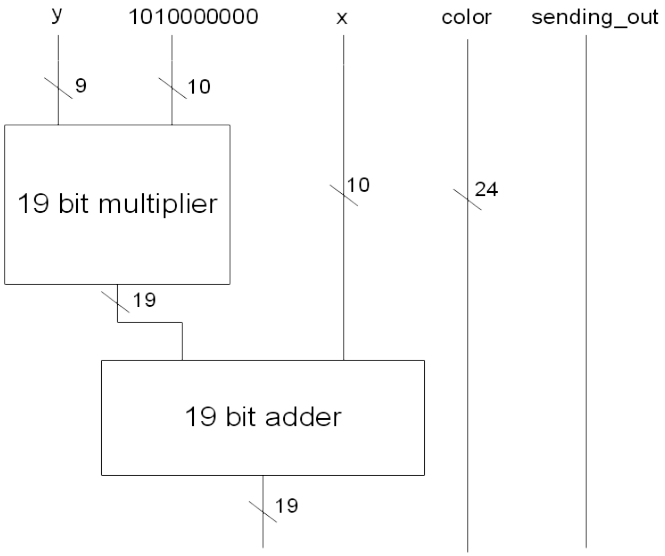
$$A = (y \cdot x_{max}) + x \tag{4}$$

Pri zvolenej konfigurácii je možné nahradiť x_{max} konštantou, jediné podporované rozlíšenie má dĺžku riadku 640 bodov. Po dosadení a pridaní rozsahov sa získa vzorec:

$$A[18:0] = (y[8:0] \cdot 640) + x[9:0] \tag{5}$$

Z tohto vzorca vyplýva, že modul sa skladá z jednej 19 bitovej násobičky a jednej 19 bitovej sčítačky s redukovaným vstupom. Ich zapojenie je zobrazené na obr. 4.5. Pri implementácii je použitá násobička zabudovaná priamo v FPGA čipe Spartan 3.

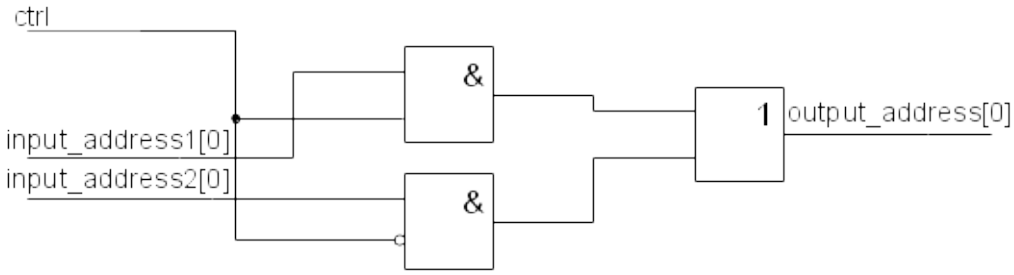
Vytvorenie samostatného modulu pre násobenie a sčítavanie by bolo neefektívne, väčšina FPGA čipov má zabudované vysoko optimalizované bloky pre aritmetické operácie.



Obr. 4.5: Interné zapojenie modulu co2add

Po konverzií sú dáta predané na zápis do vstupnej medzipamäte. Konvertor je zdieľaný medzi akceleračnými modulmi, systém nepodporuje paralelné spracovanie príkazov, takže ku konfliktom na vstupoch modulu by nemalo dôjsť. Signály *color* a *sending_out* sú predané bez zmeny.

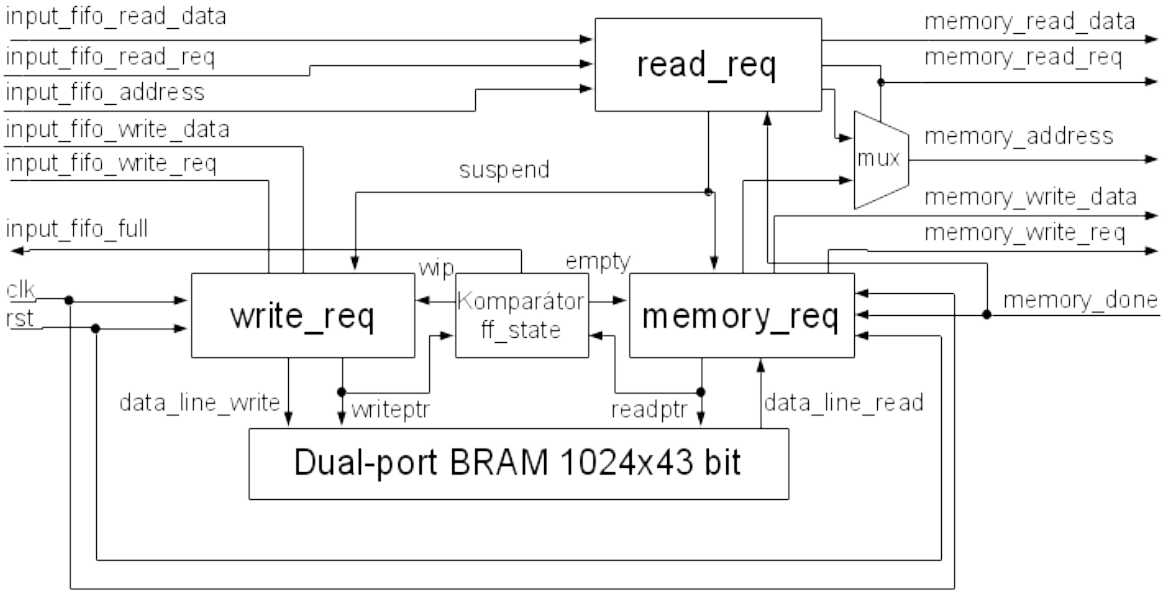
4.9 Multiplexor input_fifo_mux



Obr. 4.6: Vnútorná štruktúra jedného bitu multiplexoru modulu input_fifo_mux

Modul prepája na základe ovládacieho signálu z registrov výstupy z konvertoru co2add alebo z registrov na vstup medzipamäte *input_fifo*. Dáta sú posielané do vstupnej medzipamäte priamo, bez ďalšieho spracovania. Komponent sa skladá z 43-bitového multiplexoru typu 2-to-1 s jedným kontrolným signálom. Na obr. 4.6 je možné vidieť schému na hradlovej úrovni jedného z multiplexorov. Vstupné a výstupné dáta sa skladajú z 19 bitovej adresy zapisovaného pixelu a jeho farebnej hĺbky s rozsahom 24 bitov.

4.10 Medzipamäť input_fifo



Obr. 4.7: Bloková schéma modulu input_fifo

Vstupná medzipamäť *input_fifo* je použitá primárne na zníženie potencionálne vysokej latencie navrhovaného obvodu, spojenej s blokovaním zobrazovacej pamäte počas periodicky sa opakujúcich zobrazovacích cyklov. Modul využíva pamäť BRAM poskytovanú čipom FPGA. Maximálny počet záznamov je 1024, pričom každý záznam má veľkosť 49 bitov a obsahuje údaje o adrese a farbe zapisovaného pixelu. Celková veľkosť pamäte je 50176 bitov. Do pamäte sú ukladané iba požiadavky na zápis, čítanie prebieha prednostne. Vzhľadom na možnosti použitej zbernice by malo využitie medzipamäte pri čítaní negatívny vplyv na rýchlosť odozvy systému.

Bloková schéma interného zapojenia ovládacej logiky *input_fifo* je na obrázku obr. 4.7. Hlavným komponentom sú dva registre slúžiace na adresovanie čítania *readptr* a zápisu *writeptr*, ktoré sa postupne inkrementuju o 1 po každej operácií. Na základe porovnania uložených hodnôt v týchto registroch je určené zaplnenie medzipamäte a sú generované stavové signály *full* – signalizácia zaplnenia pamäte, *empty* – pamäť prázdna a *wip* – pozastavuje prácu. Nezávisle je generovaný signál *suspend* ktorý pozastavuje prácu celého modulu počas čítania.

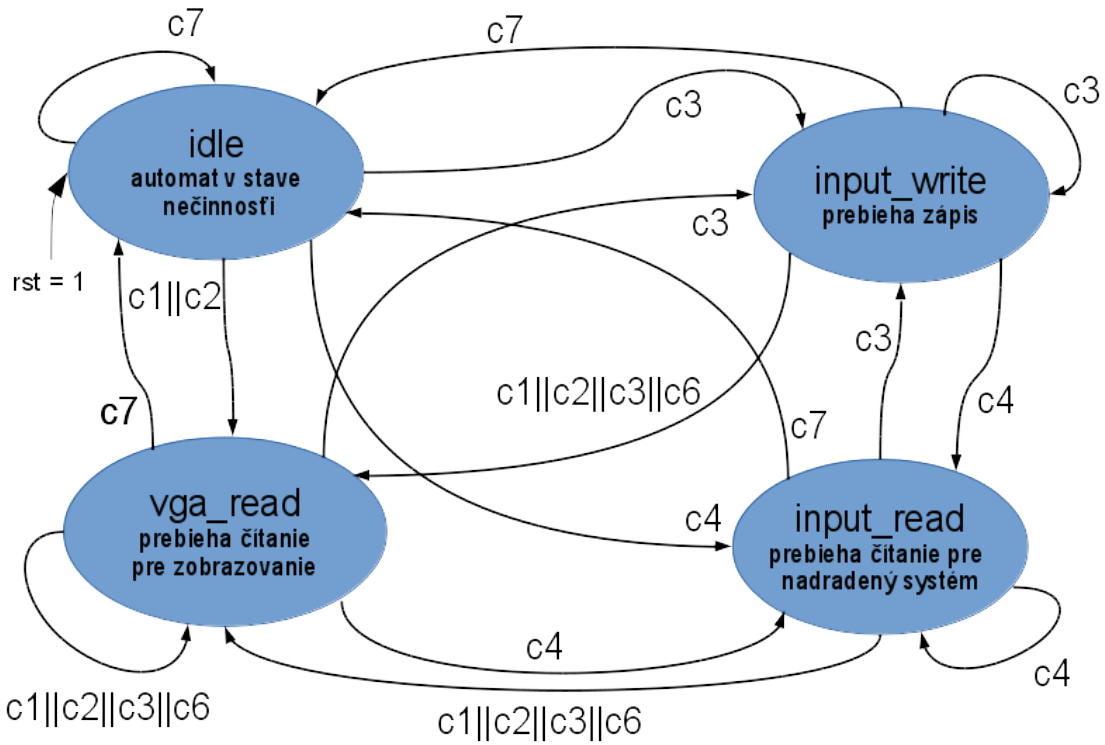
Príkazy na čítanie a zápis z nadradeného modulu, pri súčasnej architektúre registrov sú riešené v samostatných procesoch. Odosielanie dát do kontroléru pamäte je začaté v momente keď je uložený aspoň jeden záznam a signál *empty* je nastavený na logickú 0.

4.11 Komponent memory_ctl

Najkomplexnejší modul v grafickom jadre je radič pamäte *memory_ctl*. Modul prijíma dáta, požiadavky na zápis alebo čítanie a kontrolné signály z medzipamätí *vga_fifo*, *input_fifo* a kontroléra *vga_driver*. Úlohou *memory_ctl* je spracovať tieto vstupy tak aby

nedošlo ku konfliktom a na základe komunikácie so SRAM ovládačom prideľovať prístupový čas pre manipuláciu so zobrazovacou pamäťou a predávať dáta jednotlivým modulom. Modul sa skladá z registrovej časti a rozsiahleho stavového automatu. Registrová časť obsahuje pomocné registre pre zápis do pamäte alebo čítanie z pamäte začaté nadradeným systémom a čítanie pre VGA počas zobrazovania.

Stavový automat je popísaný štyrmi stavmi, v ktorých prebiehajú jednotlivé operácie a rozhodovanie o priorite vykonávania budúcich požiadaviek. Tieto stavy sú: pokojový stav *idle*, sekvenčný zápis *input_write*, *vga_read*, v ktorom prebieha čítanie dát určených na zobrazenie a stav *input_read*, ktorý nastavuje potrebné signály pri čítaní z pamäte pre nadradený systém. Prechody medzi stavmi sú zobrazené na obr. 4.8, podmienky sú označené ako c1-c7 a kombinácie vstupných a interných signálov pre ich splnenie sa nachádzajú v tab. 4.1.



Obr. 4.8: Prechody medzi jednotlivými stavmi stavového automatu modulu memory_ctl

Tab. 4.1: Kombinácie signálov pre jednotlivé podmienky prechodu medzi stavmi

Podmienka	vga_sync	vga_line_sync	vga_fifo_low	input_read_req	input_write_req	rip
c1	1	-	-	-	-	-
c2	0	1	-	-	-	-
c3	0	0	1	-	-	-
c4	0	0	0	1	-	-
c5	0	0	0	0	1	-
c6	0	0	0	0	0	1
c7	0	0	0	0	0	0

Stav *idle* je pokojový a základný stav, do ktorého systém prechádza v prípade resetu alebo počas časových úsekov v ktorých neprebiehajú žiadne operácie s pamäťou. Prechod do iného stavu nastáva príchodom požiadaviek z pripojených modulov nasledujúcim spôsobom:

Signál *vga_sync* má najvyššiu prioritu, všetky registre sú vynulované a po prečítaní prvej hodnoty z adresy pamäte 0x00000 je začaté zapĺňanie VGA medzipamäte prepnutím do stavu *vga_read*. Paralelne k tomu začína čítanie zo SRAM a je nastavený signál signalizujúci nový riadok.

V prípade synchronizačného impulzu *vga_line_sync* sa systém chová podobne ako pri *vga_sync* ale *vga_address_reg* register nie je nastavený na nulovú hodnotu, vzhľadom ku tomu že uchováva informáciu o momentálnej polohe posledného zobrazeného bodu. Stavový automat je opäť nastavený do stavu *vga_read*. Systém nemôže prejsť do stavu *idle* počas čítania pre zobrazovanie dát a toto zobrazovanie je iniciované výhradne pomocou synchronizačných signálov VGA kontrolér. Nie je potrebné definovať dodatočné prechody medzi *idle* a *vga_read*.

Ďalšie udalosti, ktoré ukončujú stav *idle* sú príkazy z nadradeného systému, konkrétne zápis dát reprezentujúcich pixel a čítanie zo zadanej adresy. Čítanie má v tomto prípade prednosť, vzhľadom na spôsob fungovania použitej komunikačnej zbernice a redukciu latencie systému pri zápise použitím vstupnej medzipamäte. Jediný potencionálny problém, ktorý môže nastať, je ak požiadavka na čítanie, prišla neskoršie ako požiadavka na zápis s rovnakou adresou a nové dáta sú ešte uložené v medzipamäti. Vtedy sa v odpovedi budú nachádzať staré dáta. Táto skutočnosť musí byť zohľadnená pri tvorbe ovládača a používaní grafického jadra. Po prijatí príkazu na čítanie je uložená vstupná adresa do registra a spolu so signálom začatia čítania *read_req* odoslaná ovládaču SRAM. Nasledujúci stav je nastavený na *input_read*.

Najnižšiu prioritu má požiadavka na zápis dát. V prípade, že neprebiehajú žiadne ďalšie podnety z pripojených modulov, je nastavený budúci stav stavového automatu na *input_write*. Zároveň sú dáta a adresa zápisu načítané do registrov a signál požiadavky

na zápis *write_req* je nastavený na úroveň logickej 1.

Počas všetkých zmien stavu modul ohlasuje okoliu že je zaneprázdnený pomocou signálu *system_ready* a neprijíma ďalšie príkazy až do ukončenia vykonávanej operácie.

Vo stave *input_write* prebieha zápis do zobrazovacej pamäte. Zápis môže byt započatý iba modulom *input_fifo,* ktorý slúži ako medzipamäť pre dáta určené na zápis a adresy miesta kde majú byť zapísané. V základe, počas trvania tohto stavu systém čaká na potvrdenie ukončenia prebiehajúcej operácie zápisu z ovládača *sram_driver*. Adresa, dáta nesúce informácie o farbe a signál vyžadovania zápisu *write_req* sú stabilne nastavené a odosielané cez výstupy registrov. Počas čakania systém nereaguje na externé podnety, s výnimkou synchronizačných pulzov *vga_sync* a *vga_line sync*, ktoré musia byť za každých okolností registrované a vzhľadom na ich krátke trvanie by nemuselo dôjsť ku ich zachyteniu. Ak sa vyskytnú počas zaneprázdnenia systému, registre *sync_req* a *sync_line_req* sú nastavené na úroveň 1. Výstup týchto registrov je ekvivalentný samotným synchronizačným signálom v momente spracovania, preto musia byť okamžite resetované na nulový výstup. V prípade dokončenia prebiehajúcej operácie je vyhodnotené nasledovné chovanie systému na základe vstupov z okolitých modulov a interných stavových signálov.

Ak systém zaregistroval synchronizačné pulzy, dochádza k vynulovaniu registrov používaných pri čítaní z pamäte pre VGA výstup a je nastavený stav *vga_read*, v ktorom je započaté zapĺňanie VGA medzipamäte. Rozdiel medzi synchronizáciou riadku a celej obrazovky je v chovaní registra *vga_address_reg*, ktorý je vynulovaný iba v prípade pulzu *vga_sync*. Tieto operácie majú najvyššiu prioritu, aby bolo zabezpečené korektné zobrazovanie za každých okolností. Nesmie nastať stav, kedy dáta nasledujúceho pixelu nie sú uložené v medzipamäti v momente, keď príde požiadavka z VGA kontroléra. Okrem toho vynechanie synchronizačného pulzu by mohlo znamenať ďalšie chyby zobrazovania ako posunutie obrazu alebo jeho skreslenie. Vzhľadom na vyššie uvedenú nutnosť mať dostatok dát v medzipamäti, má najvyššiu prioritu, s výnimkou synchronizačných signálov, čítanie pre zobrazovanie obrazu cez VGA port, keď je hlásený nedostatočný stav zaplnenia pamäte *vga_fifo*. To je vykonané prostredníctvom vstupného signálu *vga_fifo_low*. Po prijatí tejto žiadosti sa dokončí posledný prebiehajúci zápis a stavový automat prechádza do stavu *vga_read*. Okrem toho je nastavený register neukončeného čítania *rip_reg* a systém sa hlási ako zaneprázdnený.

V prípade že neprebieha synchronizácia alebo *vga_fifo_low* je nastavený na logickú 0, má prednosť dokončovanie posledného zápisu sekvencie zapisovaných pixelov. Táto funkcia bola pridaná aby sa zistilo správne vyprázdnenie registrov pre zápis *write_data_reg, write_address_reg* a nedochádzalo ku konfliktom pri prechode medzi operáciami, prípadne vynechaniu zapisovaných dát. V súlade so zvyškom systému má pri príkazoch nadradeného systému štandardné čítanie prednosť pred štandardným zápisom. V prípade čítania sú nezapísané hodnoty uložené do registrov a zápis je pozastavený,

až do ukončenia čítania. Na výstup je nastavená adresa z ktorej majú byť načítané dáta a nasledujúci stav je zmenený na *input_read*. V prípade že neprebiehajú žiadne vonkajšie podnety s vyššou prioritou a modul *input_fifo* stále hlási prítomnosť dát určených na zápis, stavový automat zotrváva v stave zápisu a systematicky načítava a odosiela nové dáta do ovládača pre SRAM *sram_driver*, až kým nedôjde k ukončeniu zápisovej sekvencie. Najnižšiu prioritu má čítanie dát na zobrazovanie z pamäte bez dodatočných signálov vyžadujúcich okamžitú obsluhu. V tomto prípade je opäť zmenený stav na *vga_read* a systém postupuje rovnako ako v prípade hlásenia *vga_fifo_low*. Ak nie je hlásená žiadna nová požiadavka z okolitých modulov alebo neprebieha žiadna interná operácia systém prechádza do pokojového stavu *idle* a všetky nastavené registre s výnimkou *vga_address_reg* sú vynulované, aby sa predišlo problémom s pretečením registrov, alebo predávaním nesprávnych hodnôt.

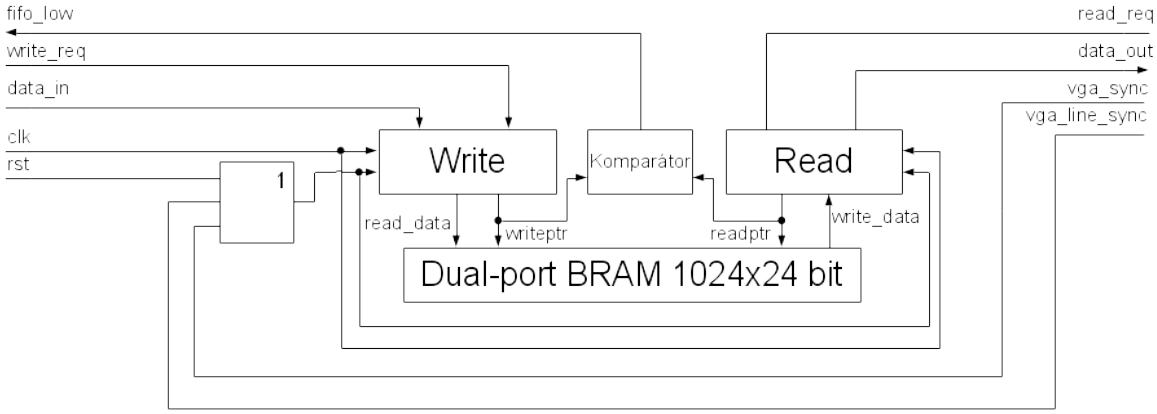
Čítanie prebieha počas stavu *input_read*. Operácia sa skladá sa z troch krokov: odoslanie adresy a signálu *read_req* do SRAM kontroléra, čakanie na dáta a poslanie prijatých dát nadradenému systému. Počas čakania na dáta je systém uzamknutý v danom stave a neprijíma ďalšie požiadavky. Prechody medzi stavmi sú v tomto prípade riešené na rovnakom princípe ako v stave *input_write,* s výnimkou dokončovania sekvencie operácií zápisu, čo môže nastať iba počas *input_write.* Samotné priraďovanie signálov je upravené pre odlišné potreby stavu *input_read.* Opäť platí že najvyššiu prioritu majú synchronizačné pulzy, potom zaistenie bezproblémového predávania dát na zobrazovanie, čítanie, zápis a ak neexistuje žiadna iná požiadavka, doplnenie VGA medzipamäte do konca riadku. Nakoniec, ak po ukončení posledného čítania systém neregistruje žiadne externé príkazy, vracia sa do stavu *idle*, v ktorom čaká na nové podnety.

Posledný stav, popísaný v stavovom automate modulu *memory_ctl,* je *vga_read*. Tento stav sa stará o dodávanie korektných dát na zobrazovanie. Identifikácia a čítanie správnych pixelov je riešená iba v tomto module, ďalej sa nekontroluje a ostatné moduly automaticky predpokladajú že prijímajú správne dáta. Odosielanie prebieha po cykloch dlhých 640 bodov, čo je dĺžka jedného riadku. Adresa je postupne inkrementovaná, spolu s registrom umiestnenia v riadku *line_reg*. Synchronizácia s ostatnými modulmi systému prebieha pomocou synchronizačných signálov *vga_line_sync* a *vga_sync*, ktoré označujú koniec riadku, respektíve ukončenie zobrazenia celej obrazovky a začatie čítania znovu z adresy 0x00000. Rovnako ako pri ostatných operáciách, systém čaká na potvrdenie dokončenia zo SRAM ovládača a medzitým ignoruje všetky externé podnety, okrem synchronizácie, ktorá musí byť za každých okolností registrovaná. Samotné čítanie prebieha rovnako ako v stave *input_read*, akurát dáta sú odoslané do medzipamäte *vga_fifo* a adresa je nastavovaná interne.

Prechody medzi stavmi sú čiastočne analogické k ostatným stavom. V prípade synchronizačných impulzov, systém zotrváva v stave *vga_read*, ale príslušné registre sú nastavené na nulovú hodnotu. Platí že signál *vga_sync* má vyššiu prioritu ako

vga_line_sync. V prípade že *vga_fifo* má nastavený signál *vga_fifo_low* na logickú 1, požiadavky z nadradeného systému sú ignorované, až kým nie je medzipamäť dostatočne zaplnená. Ďalšie operácie sú uprednostňované v nasledujúcom poradí: čítanie pre nadradený systém, zápis, čítanie pre zobrazovací cyklus a ak sa nevyskytujú žiadne požiadavky stavový automat prechádza do stavu *idle*. Zápis do medzipamäte prebieha počas jedného hodinového cyklu, bez potvrdzovacieho signálu. Modul predpokladá úspešný zápis behom jednej periódy, iné chovanie by znamenalo, vzhľadom na architektúru grafického jadra závažnú internú chybu systému.

4.12 Medzipamäť vga_fifo



Obr. 4.9: Bloková schéma modulu vga_fifo

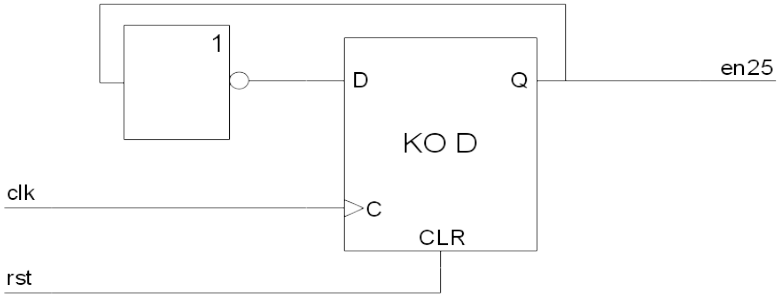
Podobne ako modul *input_fifo* je *vga_fifo* medzipamäť určená primárne na zlepšenie výkonu grafického jadra a minimalizovanie dopadu synchrónneho čítania blokov pamäti pri zobrazovaní. Základ modulu tvorí pamäť o veľkosti 1024 záznamov s veľkosťou 24 bitov, implementovaná ako interná BRAM čipu FPGA. Reálne je využívaných iba 640 záznamov, vyššia hodnota by nepriniesla žiadne výhody a syntetizátor automaticky zarovnáva adresný priestor na najbližší bit.

Modul *vga_fifo* sa chová ako pamäť typu FIFO, prvé uložené dáta sú aj ako prvé odoslané. To prebieha prostredníctvom ukazovateľov, ktoré sú pri každej operácii inkrementované o 1, čím ukazujú na novú adresu v pamäti. O vstupy pri zápise sa stará modul *memory_ctl* a uložené informácie určené na zobrazenie sú odosielané do modulu *vga_driver*. Interná štruktúra je zobrazená na obr. 4.9.

Riadiace signály sa skladajú zo synchronizačných pulzov vga_sync, respektíve vga_line_sync, ktoré nastavujú interné ukazovatele na nulovú adresu, čím sa pamäť chová ako prázdna a signálov požiadaviek *write_req* a *read_req*. Synchronizačné signály fungujú ako synchrónny reset modulu. Modul nezasiela potvrdenia ukončenia operácií, tie musia byť vždy ukončené do poslednej požiadavky, v priebehu jednej periódy hodinového cyklu. Na hlásenie nedostatočného stavu počtu záznamov uložených v pamäti a iniciáciu zápisu

zo strany *memory_ctl* modulu slúži signál *vga_fifo_low*. Tento signál sa prepne do úrovne 0, ak je prebytok aspoň 15 záznamov a naopak do úrovne 1, ak je prebytok 10 záznamov a menej alebo v prípade resetu. Tento rozdiel slúži na zlepšenie podpory sekvenčných operácií. Zmena stavu nenastáva v priebehu minimálne desiatich períód hodinového signálu.

4.13 Delička hodinového signálu clk_divider



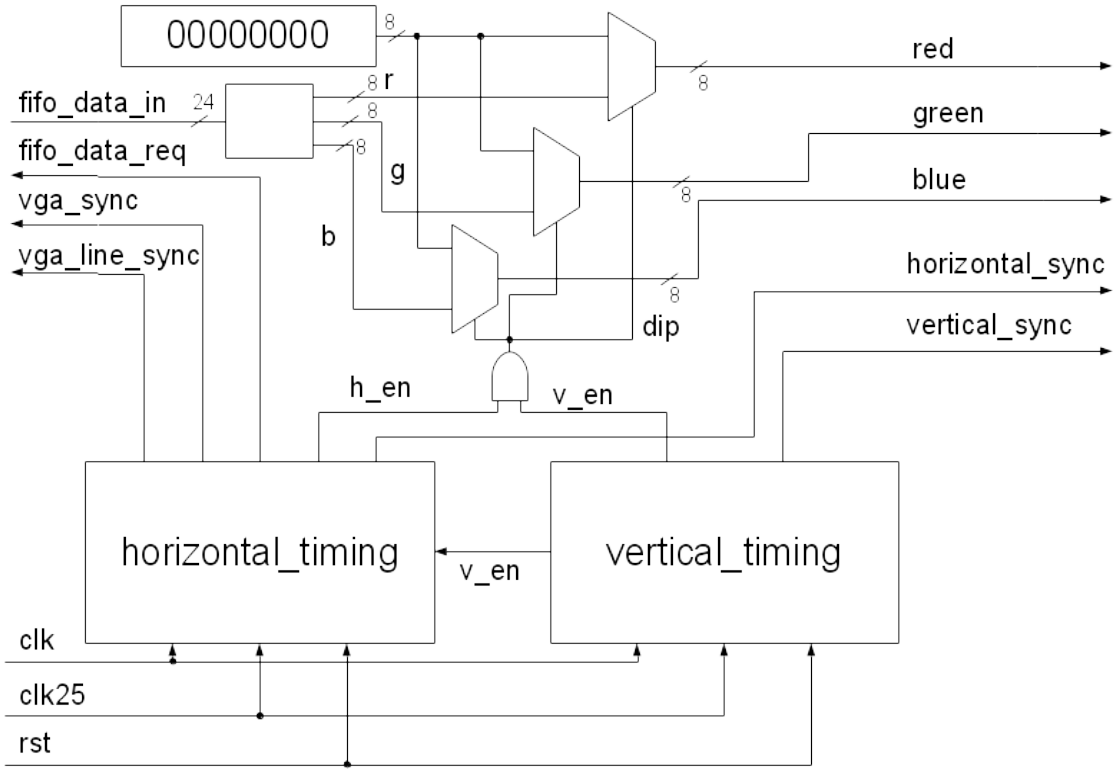
Obr. 4.10: RTL schéma deličky clk_divider

Pre správne časovanie synchronizačných signálov vertikálnej a horizontálnej synchronizácie potrebuje VGA kontrolér *vga_driver* pracovať na frekvencii hodinového signálu 25 MHz. V implementácií to je riešené použitím frekvenčného deliča vstupného hodinového signálu. Ten zníži frekvenciu o polovinu. Modul sa skladá z invertoru a preklápacieho obvodu typu D. Schému je možné vidieť na obr. 4.10. Na každej vzostupnej hrane hodinového signálu sa výstup zmení na opačnú hodnotu, tým vznikne signál s dvojnásobne dlhšou periódou ako originálny signál. Výhody tohto riešenia oproti využitiu DCM bloku FPGA Spartan 3, určeného na generovanie hodinových signálov rôznych frekvencií, sú nízka hardvérová náročnosť a nezávislosť na IP blokoch špecifických pre konkrétny čip. Nevýhodou je nemožnosť použiť v obvode inú frekvenciu hodinového signálu, bez zmeny hardvérovej konfigurácie alebo zavedenia problémov s časovaním VGA výstupu.

4.14 Kontrolér vga_driver

Modul *vga_driver* poskytuje podporu pre rozhranie VGA portu. Časovanie je prispôsobené pre rozlíšenie 640×480 s obnovovacou frekvenciou 60 Hz. Celý modul pracuje na frekvencií 25 MHz. Pre správnu funkciu vyžaduje *vga_driver* hodinový signál *clk* s frekvenciou minimálne 50 MHz, a 25 MHz povoľovací signál *en25*. Časovacie parametre synchronizačných signálov *h_sync* a *v_sync* sú zadané v generic časti VHDL entity, takže zmeny môžu byť vykonané relatívne jednoducho, ale zmena vo funkčnom režime je nemožná. Je nutné obvod opätovne syntetizovať, vygenerovať programovací súbor a nahrať ho do FPGA. Modul sa skladá zo 4 častí – sekvenčnej časti obsahujúcej registre, jednoduchých kombinačných obvodov pre obsluhu vstupov a výstupov modulov

a procesov na generovanie synchronizačných a riadiacich signálov. Bloková schéma modulu je na obr. 4.11.



Obr. 4.11: Bloková schéma modulu vga_driver

Proces *horizontal_timing* slúži na tvorbu signálov potrebných na správnu horizontálnu synchronizáciu obrazu. Jeho primárny komponent je 11 bitový čítač, ktorý sa cyklicky inkrementuje až po hodnotu reprezentujúcu koniec riadku. Táto hodnota je súčet hodinových periód všetkých častí synchronizačného cyklu. Na základe hodnoty uloženej v registri čítača a konštánt pre zvolené časovanie obrazu sú priraďované hodnoty signálom *h_en*, *h_sync*, *pix_req* a *vga_line_sync*.

Signál *h_en* reprezentuje dobu zobrazovania na monitore. Je použitý v kombinácií so signálom *v_en* generovaným procesom na vertikálnu synchronizáciu. Ak je výsledok logickej operácie and medzi *v_en* a *h_en* rovný logickej 1, prebieha zobrazovanie obrazu. Horizontálna synchronizácia je tvorená signálom *h_sync*, ktorý je priamo pripojený na príslušný výstup VGA kontroléra *horizontal_sync*. Vzhľadom na oneskorenie práce registrov minimálne o jeden hodinový signál pri komunikácií so zvyškom systému je nutné odosielať požiadavky na čítanie o jednu periódu hodinového signálu skôr ako je započaté zobrazovanie. Táto požiadavka je reprezentovaná signálom *pix_req*. Posledný spomenutý signál *vga_line_sync* je generovaný na jednu periódu hodinového signálu pri ukončení zobrazovania riadku. Jedná sa o synchronizačný impulz pre radič pamäte *memory_ctl* a VGA medzipamäť *vga_fifo*, slúžiaci na správnu synchronizáciu čítania z pamäte vzhľadom na zobrazovaný obraz u týchto modulov a ochranu proti problémom

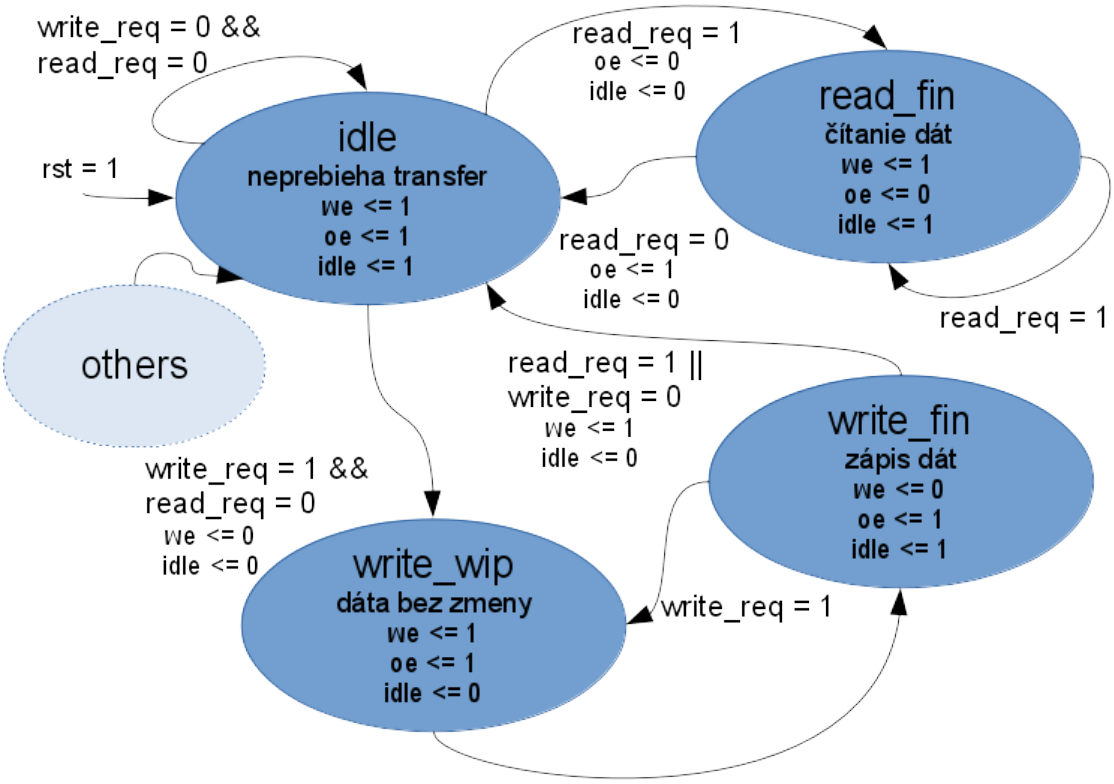
s časovaním.

Nezávisle na horizontálnom časovaní je v procese *vertical_timing* tvorené vertikálne časovanie. Oddelenie vertikálnej a horizontálnej synchronizácie je definované v popise rozhrania VGA portu. Základ procesu opäť tvorí čítač, tentokrát s 20 bitovým rozsahom. Tento čítač pripočítava v každom hodinovom cykle jednotku ku svojej hodnote, až kým nedosiahne posledný bod cyklu vertikálnej synchronizácie, keď sú všetky bity čítača nastavené na nulu a cyklus sa dostáva na začiatok. Analogicky ku signálom v procese *horizontal_timing* sú na základe hodnoty uloženej v registri čítača a časovacích konštánt, nastavené signály *v_en*, *v_sync* a *vga_sync*. Čítanie dát jednotlivých pixelov je riešené výhradne v rámci horizontálnej synchronizácie, preto signál *pix_req* v tomto procese nemá alternatívu.

V_en vymedzuje zobrazovaciu časť vertikálneho cyklu, *v_sync* je priamo pripojený na výstup pre vertikálne časovanie VGA portu. Signál *vga_sync* signalizuje ukončenie zobrazovacieho cyklu. Jedná sa o krátky impulz zaslaný do modulov *memory_ctl* a *vga_fifo*. Úlohou tohto signálu je resetovanie čítania pre VGA zo zobrazovacej pamäte v prípade chyby a periodická synchronizácia časovania týchto modulov. V prípade neočakávaného dočasného problému pri čítaní z pamäte je nesprávne zobrazovanie zachované maximálne jednu periódu zobrazovacieho cyklu. Pri obnovovacej frekvencií 60 Hz to znamená maximálny čas nekorektného zobrazenia 16,667 ms. Samozrejme ak sa jedná o závažnú, opakovane sa vyskytujúcu chybu, alebo nesprávny zápis do zobrazovacej pamäte, problém ostane prítomný aj po ukončení starého a začatí nového zobrazovacieho cyklu.

4.15 Kontrolér sram_driver

Jedná sa o kontrolér pamäte SRAM. Ovládač je vytvorený pre konkrétne zapojenie dvoch čipov 256k x 16 ISSI IS61LV25616AL. Modul je neprenosný na iné platformy. Implementácia je riešená pomocou sekvenčného stavového automatu a logiky prepínajúcej medzi kombinačnými signálovými cestami pri čítaní a výstupmi z registrov potrebnými pri zápise. Logika navyše ošetruje chovanie vstupno-výstupných dátových zberníc *io1* a *io2*, ktoré sú počas zápisu prepnuté do stavu vysokej impedancie, rovnako ako pri troj-stavových bunkách.



Obr. 4.12: Stavový automat modulu sram_driver s popisom základných kontrolných signálov

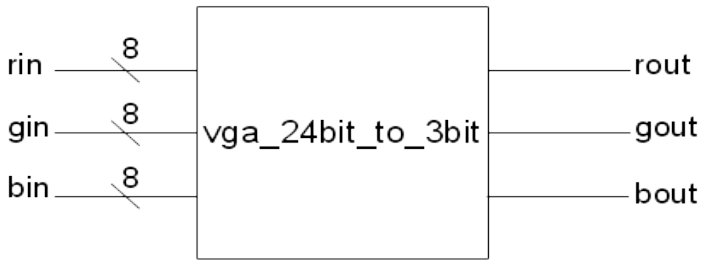
Stavový automat, znázornený na obr. 4.12, sa stará o správne zapisovacie a čítacie sekvencie, multiplexovanie signálov *ce1, ce2* na základe adresy a riešenie konfliktov pri prepínaní medzi jednotlivými operáciami.

Zápis prebieha v 2 stavoch: *write_fin* v ktorom sú nastavené nové hodnoty, prípadne dochádza ku prepínaniu medzi operáciami a *write_wip* v ktorom je dokončený *we* impulz, pri udržiavaných hodnotách v registroch.

Čítanie prebieha počas jedného hodinového cyklu v stave *read_fin*. Na každej nábežnej hrane signálu *clk* sú zapísané parametre novej požiadavky a prečítané dáta odoslané sú do grafického jadra na ďalšie spracovanie.

Stav *idle* je základný stav, v ktorom sú registre vynulované a prebieha obsluha prvých požiadaviek pri sekvencií viacerých operácií v rade. Stavový automat do neho prechádza, ak je ukončená sekvencia požiadaviek alebo došlo k resetu obvodu alebo zakázanému stavu. Všetky registre s výnimkou *vga_address_reg*, obsahujúceho informáciu o adrese nasledujúceho bodu prečítaného zo zobrazovacej pamäte, sú vynulované. Register *vga_address_reg* môže byť vynulovaný iba asynchrónnym resetom alebo pulzom signálu *vga_sync* generovaným VGA kontrolérom po skončení zobrazovacej fázy vertikálnej synchronizácie. Prechod do iného stavu prebieha pomocou signálov požiadaviek alebo synchronizačných signálov, pričom je jasne stanovená priorita, v prípade viacerých požiadaviek počas jednej periódy hodinového signálu.

4.16 Konvertor farieb



Obr. 4.13: Modul vga_24bit_to_3bit a jeho vstupy/výstupy

Vga_24bit_to_3bit je špecifický modul vyžadovaný testovacím zapojením. Vstupné a výstupné signály so zobrazené na obr. 4.13. Jeho jediná funkcia je konverzia 24 bitovej farebnej hĺbky na trojbitovú, ktorá prebieha predaním najvýznamnejšieho bitu z 8 bitových vektorov reprezentujúcich intenzitu farby a zahodením ostatných bitov. Konverzia je nevyhnutná, vzhľadom na limity VGA rozhrania použitej vývojovej dosky. Modul neobsahuje žiadnu logiku, iba zápis prepojení vstupov a výstupov, takže nepredstavuje dodatočnú záťaž pre hradlové pole FPGA.

4.17 Moduly pre podporu simulácie a testovania

Pre simuláciu bol vytvorený jednoduchý testbench. Stará sa iba o generovanie testovacích vektorov pre systém. Vzhľadom na vysokú komplikovanosť, časovú a výpočtovú náročnosť automatizovaného riešenia, prebieha vyhodnocovanie výsledkov simulácie manuálne. Testbench generuje externé signály ako hodinový signál, asynchrónny reset a simulované stlačenie tlačítka.

Tab. 4.2: Porovnanie pamäťovej náročnosti simulácie pri použití rôznych dátových typov na uloženie dát.

Typ ukladaných vektorov	Kapacita pamäte modulu(bit)	Pamäť potrebná na simuláciu(kB)
std_logic_vector	1024	81932
integer	1024	59904
std_logic_vector	8192	241508
integer	8192	69808
std_logic_vector	16384	424432
integer	16384	81138
std_logic_vector	262144	5911812
integer	262144	427172

Okrem toho je na simuláciu potrebný funkčný model pamäte. Modelovaná pamäť má identické kontrolné signály ako skutočný pamäťový čip a jej adresný rozsah nie je obmedzený. Vzhľadom na pamäťovú a výpočtovú náročnosť 9 úrovňovej logiky sú dáta ukladané ako typ integer. Porovnanie pamäťovej náročnosti bežiacej simulácie pre rôzne veľkosti celkovej pamäte sú v tab. 4.2. Tento krok zabraňuje ukladať stavy U, Z, X, W, -,

slabé logické úrovne, ale v prípade reálnej pamäte sú tieto hodnoty ukladané ako náhodný šumový signál alebo logické úrovne a výsledok reálneho zapojenia nezodpovedá simulácii. Aby bola možná komunikácia s okolitými modulmi, prebieha pri zápise aj čítaní konverzia medzi typmi integer a std_logic_vector. Časovanie pamäte je implementované behaviorálnym spôsobom. Podporované je iba základné časovanie, komplexnejší model by nepriniesol zlepšenie výsledkov, vzhľadom na zjednodušené časovanie zvyšku systému.

Každý modul reprezentuje jeden čip o veľkosti 256 kB. V simulačnom zapojení sa nachádzajú dve inštancie SRAM modelu, aby výsledok bol čo najbližšie skutočnému zapojeniu čipov.

4.18 Výsledné parametre implementovaného systému

Navrhnutý obvod bol úspešne syntetizovaný a implementovaný do cieľového FPGA. Systém vyžaduje relatívne veľký počet rôznych komponentov. Väčšina funkcií je založená na jednoduchých aritmetických operáciách, porovnávaní a využívaní rôznych typov pamäte. Návrh sa vyhýba aplikovaniu logických operácií na hodinový signál, používaní latch buniek a vytváraniu kombinačných komunikačných ciest.

Reálne využitie FPGA Spartan 3 je zobrazené v tab. 4.3, z ktorej je možné vyčítať, že navrhnutý grafický kontrolér kladie najväčšie požiadavky na blokovú RAM pamäť a počet LUT tabuliek. Maximálna rýchlosť obvodu je 50,94 MHz, čím je splnená podmienka pre použitie vstupného 50 MHz signálu. Táto hodnota je blízko limitu, preto aj malé zmeny v architektúre môžu spôsobiť problémy s nedostatočnou rýchlosťou implementovaného obvodu.

Výsledné parametre sa líšia vzhľadom na použité interné dáta v zapojení, parametre v tab 4.3 platia pre použité zdrojové kódy a bitstream test_case1, uvedený v prílohe. Tieto rozdiely sú spôsobené mierne odlišným chovaním syntetizátora XST a implementačných nástrojov pre odlišné dáta vstupné, aj v prípade rovnakého zápisu zapojenia.

Tab. 4.3: Výsledné parametre implementovaného systému.

Parameter	Celé zapojenie		Grafické jadro		Limit
	Hodnota	Využitie čipu(%)	Hodnota	Využitie čipu(%)	
KO D	1073	27,94	932	24,27	3840
LUT	2498	65,05	2236	58,23	3840
RAMB16	10	83,33	10	83,33	12
MULT18X18	1	8,33	1	8,33	12
BUFGMUX	1	12,5	1	12,5	8
IOB	74	42,77	-	-	173
Max. frekvencia	50,94 MHz	-	50,94 MHz	-	50 MHz

Záver

V tejto práci bol vytvorený návrh a implementácia grafického kontroléra pre obvody FPGA a jeho testovacieho zapojenia v rámci vývojovej dosky Spartan 3 Starter Kit Board. Pôvodne bol grafický kontrolér určený pre spoluprácu s 32 bitovým soft procesorom Codix RISC pod operačným systémom Linux, ale požiadavky tohto procesora na plochu čipu boli vyššie ako sa očakávalo, takže pre účely testovania bol vytvorený modul emulujúci nadradený systém.

Navrhnutý grafický kontrolér pracuje na frekvencii hodinového signálu 50 MHz, podporuje pevne nastavený zobrazovací mód s rozlíšením 640×480 a obnovovacou frekvenciou 60 Hz a 24 bitovou farebnou hĺbkou. Vstupné komunikačné rozhranie je zbernica CLB implementovaná v móde CLB slave, výstup prebieha cez VGA port a na komunikáciu s pamäťou je vytvorené samostatné rozhranie, vyžadujúce prídavnú medzivrstvu upravenú pre špecifické potreby konkrétnych pamäťových čipov. Podporované funkcie sú priamy prístup do zobrazovacej pamäte, umožňujúci zápis a čítanie informácií o jednotlivých pixeloch, akcelerované vykresľovanie čiar pomocou Bresenhamovho algoritmu, vykresľovanie symbolov zadaných v ROM obsahujúcej 12 bodovú verziu fontu DejaVuSans a vyplnenie obdĺžnikovej oblasti zvolenou farbou. Systém je vybavený asynchrónnym resetom.

Navrhnutý grafický kontrolér sa skladá z 8 modulov: zbernice CLB, registrov, vykresľovacej jednotky, jednotky pamäťových operácií, konvertora súradníc na adresu, multiplexora rôznych dátových vstupov, vstupnej medzipamäte, pamäťového kontroléra, deličky hodinového signálu, VGA medzipamäte a VGA kontroléra. Pre testovacie účely obvod obsahuje tri dodatočné moduly: generátor vstupných dát, konvertor farieb z 24 bitov na 3 bity a medzivrstvu pre prácu so SRAM pamäťovými čipmi.

Implementovaný grafický kontrolér je v rámci testovacieho zapojenia funkčný, vykreslené testovacie obrazce sú v prílohe.

Zoznam použitej literatúry

[1] MAINI, A. K. *Digital electronics: Principles, Devices and Applications.* Chichester: Wiley, 2007, xxiii, 727 s. ISBN 978-0-470-03214-5.

[2] BROWN, S., VRANESIC, Z. *Fundamentals of digital logic with VHDL design.* 3rd ed. New York, USA: McGraw-Hill, 2009, xx, 939 p. ISBN 00-712-8765-5.

[3] XILINX [online]. *Spartan-3 Starter Kit Board User Guide.* 2005. 64 s. [cit. 2014-05-19]. Dostupné z WWW: www.xilinx.com/support/documentation/user_guides/ug331.pdf.

[4] XILINX [online].*Spartan-3 FPGA Family Data Sheet.* 2009. 217s. [cit. 2014-05-19] Dostupné z WWW: www.xilinx.com/support/documentation/data_sheets/ds099.pdf.

[5] ISSI [online]. *IS61LV25616AL: 256K x 16 HIGH SPEED ASYNCHRONOUS CMOS STATIC RAM WITH 3.3V SUPPLY.* Milpitas, USA, 2011, 16 s.[cit. 2014-05-19]. Dostupné z: www.issi.com/WW/pdf/61LV25616AL.pdf

[6] *Codasip: ASIP Development Automation* [online]. CODASIP LTD. 2014, 19.5.2014, [cit. 2014-05-19]. Dostupné z: www.codasip.com

[7] XILINX [online]. *Command Line Tools User Guide.* 2012, 423 s. [cit. 2014-05-19]. Dostupné z:

www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/devref.pdf

[8] MENTOR GRAPHICS [online]. *ModelSim Reference Manual: Software Version 6.5e.* Wilsonville, USA, 2010, 388 s.[cit. 2014-05-19] Dostupné z: cseweb.ucsd.edu/classes/fa10/cse140L/lab/docs/modelsim_ref.pdf

[9] *VESA* [online]. VESA. 2014 [cit. 2014-05-19]. Dostupné z: http://www.vesa.org/

[10] SANCHEZ, J., CANTON M.P. *Software solutions for engineers and scientists.* Boca Raton: CRC Press, 2008, xxv, 918 p. ISBN 14-200-4302-1.

[11] IBM [online]. *IBM VGA Technical Reference Manual.* 1992, 112 s. [cit. 2014-05-19]. Dostupné z: www.mcamafia.de/pdf/ibm_vgaxga_trm2.pdf

[12] VESA. *VESA BIOS EXTENSION (VBE): Core Functions Standard*. USA, 1998, 95 s. [cit. 2014-05-19]. Dostupné z: www.petesqbsite.com/sections/tutorials/tuts/vbe3.pdf

[13] MARCHESIN, S. *Linux Graphics Drivers: an Introduction.* 2012, 69 s. [cit. 2014-05-19]. Dostupné z: http://people.freedesktop.org/~marcheu/linuxgraphicsdrivers.pdf

[14] BUELL, *A. Framebuffer HOWTO*. 2010. [cit. 2014-05-19]. Dostupné z: http://www.tldp.org/HOWTO/pdf/Framebuffer-HOWTO.pdf.

[15] BRESENHAM, J. E. *Algorithm for computer control of a digital plotter.* IBM Systems Journal[online].1965,4,1,[cit. 2014-05-19]. Dostupné z WWW: http://www.cse.iitb.ac.in/~paragc/teaching/2011/cs475/papers/bresenham_line.pdf. ISSN 0018-8670.

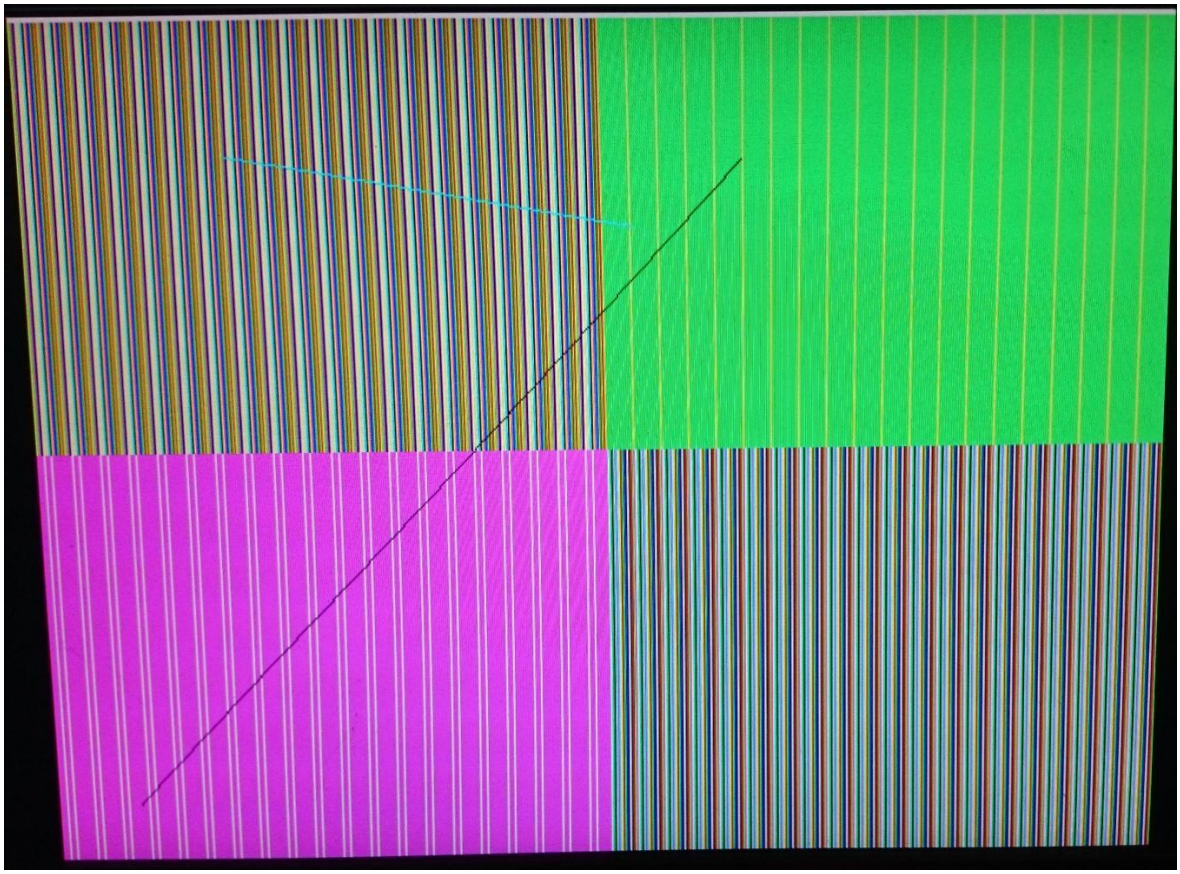
[16] *VGA Signal Timing* [online]. SECONS LTD. 2008. [cit. 2014-05-19]. Dostupné z: http://tinyvga.com/vga-timing

Zoznam skratiek

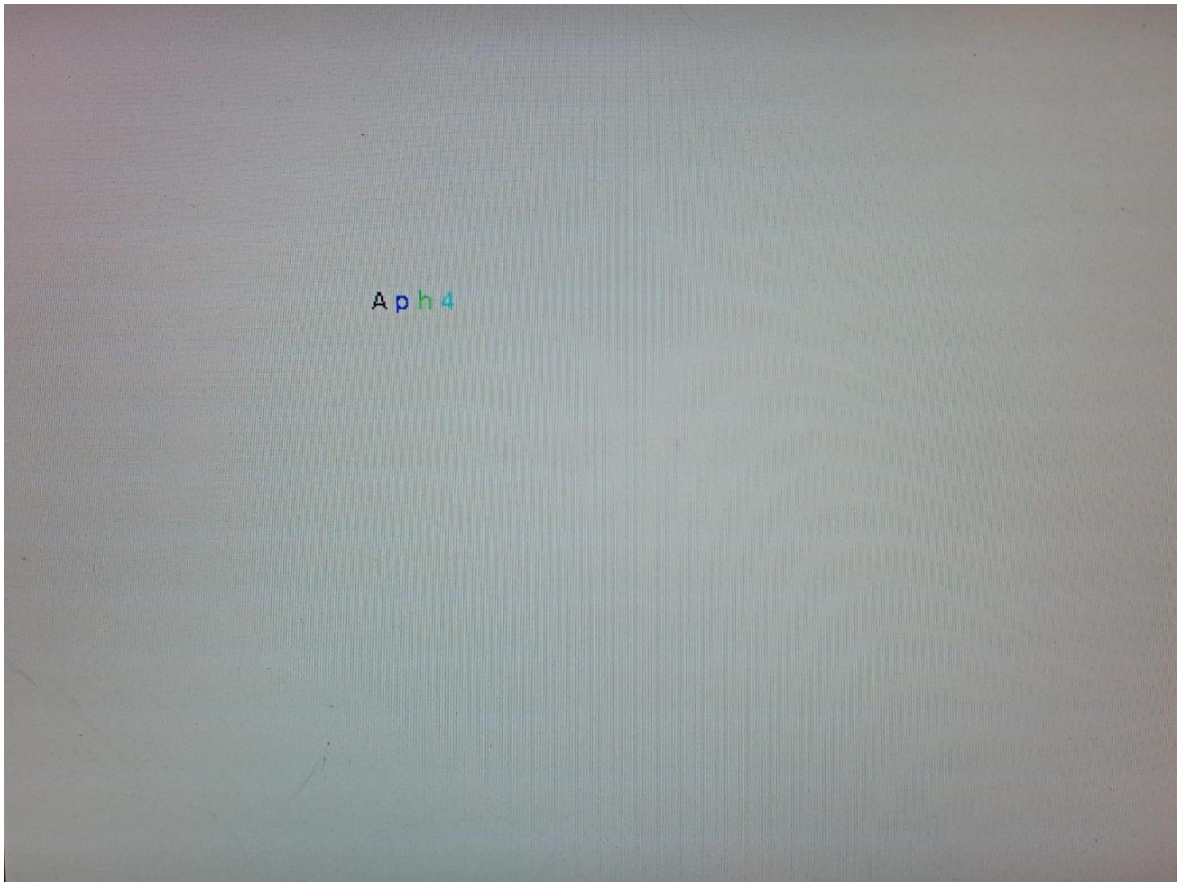
API - Application Programming Interface - rozhranie pre programovanie aplikácií
ASIC - Aplication Specific Integrated Circuit - integrovaný obvod pre špecifickú aplikáciu
BGA - Ball Grid Array - pole guličiek v mriežke
BIOS - Basic Input-Output System - základný vstupno-výstupný systém
BitBTL - Bit Boundary Block Transfer - bitovo ohraničený blokový presun
BRAM - Block RAM - bloková RAM
CLB - Codasip Local Bus - Codasip lokálna zbernica
CLB - Configurable Logic Block - konfigurovateľný logický blok
CMOS - Complementary Metal-Oxid-Semiconductor - doplňujúci sa kov-oxid-polovodič
CPLD - Complex Programmable Logic Device - komplexné zariadenie s programovateľnou logikou
DRM - Direct Rendering Manager - správca priameho renderingu
E-EDID - Enhanced Extended Display Identification Data - vylepšené rozšírené identifikačné dáta displeju
Fbcon – Framebuffer console – grafická konzola
Fbdev – Framebuffer device – zobrazovacie zariadenie
FIFO - First In First Out - prvý dovnútra, prvý von
FPGA - Field Programmable Gate Array - programovatelné hradlové pole
HDL - Hardware Description Language - jazyk popisujúci hardvér
JTAG - Joint Test Action Group - spojená skupina pre testovacie akcie
LED - Light Emmiting Diode - dióda emitujúca svetlo
LUT - Look-Up Table - náhľadová tabuľka
PROM - programmable ROM - programovatelná ROM
RAM - Random Access Memory - panäť s náhodným prístupom
RAMDAC - RAM Digital-to-Analog Converter - D/A prevodník využívajúci RAMDAC
RGB - Red Green Blue - červená zelená modrá
ROM - Read Only Memory - pamäť iba pre čítanie
RTL - Register Transfer Level – úroveň prenosu registrov
SRAM - Static RAM - statická RAM
SVGA - Super VGA - super VGA
UCF - User Constraint File - súbor s uživatelsky definovanými nastaveniami
VBE - Vesa BIOS Extensions - VESA rozšírenia pre BIOS
VESA - Video Electronics Standards Asociation - Asociácia pre elektronické video štandardy
VGA - Video Graphics Array - video-grafické pole
VHDL - VHSIC Hardware Description Language - jazyk popisujúci VHSIC hardvér
VHSIC - Very High Speed Integrated Circuit - vysokorýchlostný integrovaný obvod

Zoznam príloh

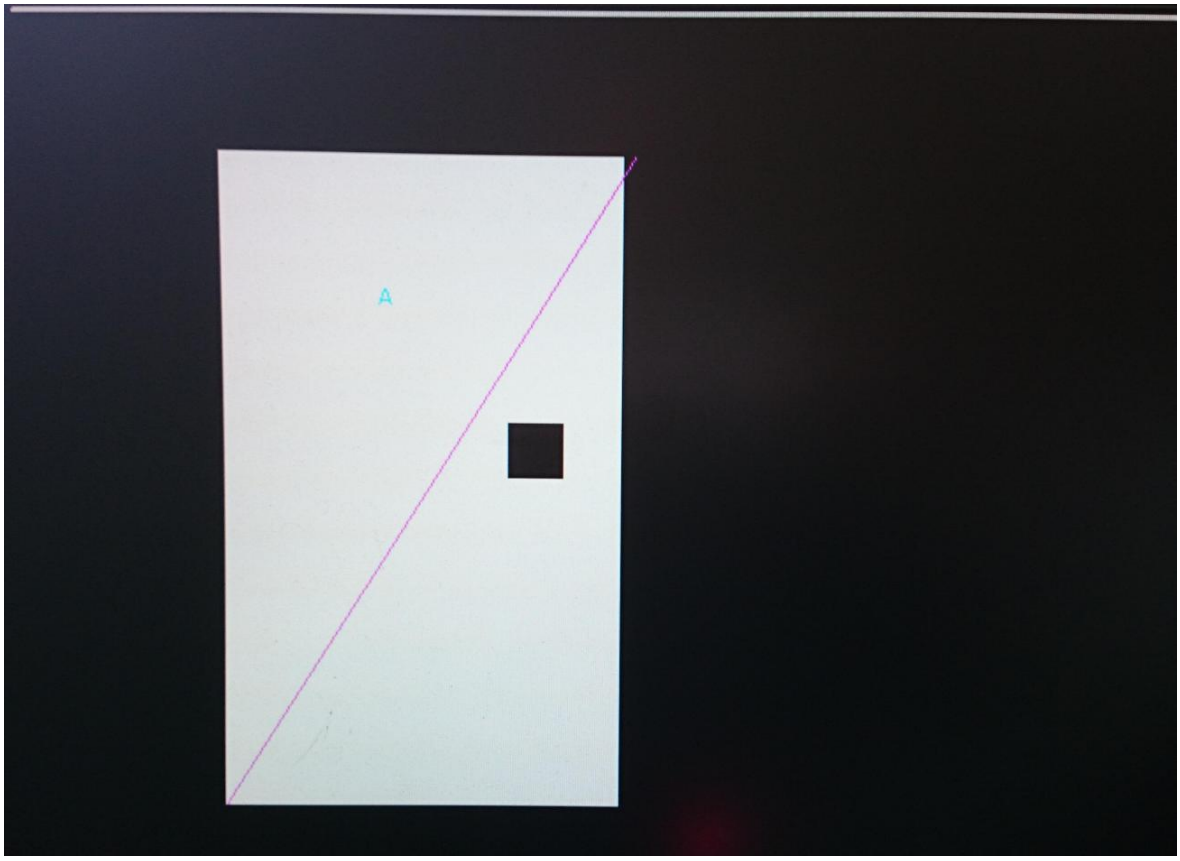
Príloha 1.1 Zobrazenie testovacieho obrazcu 1



Príloha 1.2 Zobrazenie testovacieho obrazcu 2



Príloha 1.3 Zobrazenie testovacieho obrazcu 3



Príloha 1.4 Zobrazenie testovacieho obrazcu 4